GRAZ UNIVERSITY OF TECHNOLOGY

Hash Function Cryptanalysis

Vincent Rijmen

Institute for Applied Information Processing and Communications (IAIK) - Krypto Group

Faculty of Computer Science
Graz University of Technology

Thanks to N. Pramstaller and C. Rechberger

Overview

- **Intro**
- Outline of SHA-0/SHA-1
- Generating a "small" collision
- Extensions to full collisions
- SHA-1
- Concluding remarks

Old approaches

- Several people have shown weaknesses in the compression function of MD4.
  - Bosselaers and den Boer
  - Merkle
  - Vaudenay
  - Dobbertin
- Dobbertin's techniques received the most attention in the scientific community.
  - Lead to a fast collision-producing algorithm
  - Even meaningful collisions can be produced

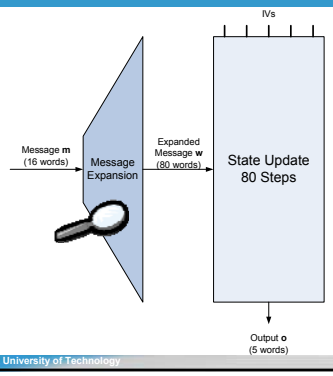
Old approaches

- Dobbertin also sketched an attack on MD5
  - Gets increasingly complicated
  - No hope for SHA-1

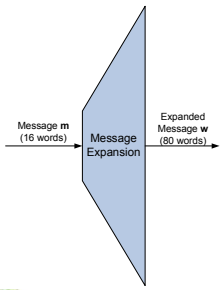
New approaches: SHA

- SHA-0
  - Chabaud and Joux [98]
  - Biham and Chen [04]
  - Joux et al. [04]
  - Wang et al. [05]
- SHA-1
  - Biham and Chen [04]
  - Rijmen and Oswald [05]
  - Wang et al. [05]

Outline of SHA



Outline of SHA – Message Expansion



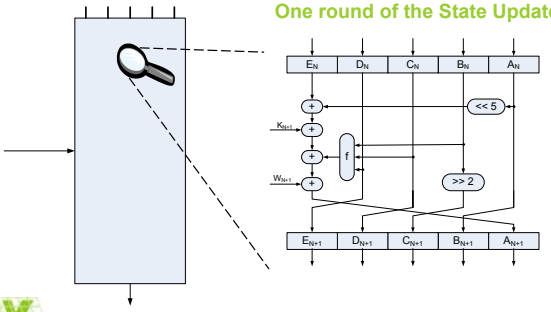
SHA-0

$$W_t = \begin{cases} M_t & for\,(0 \le t \le 15) \\ W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16} & for\,(16 \le t \le 79) \end{cases}$$

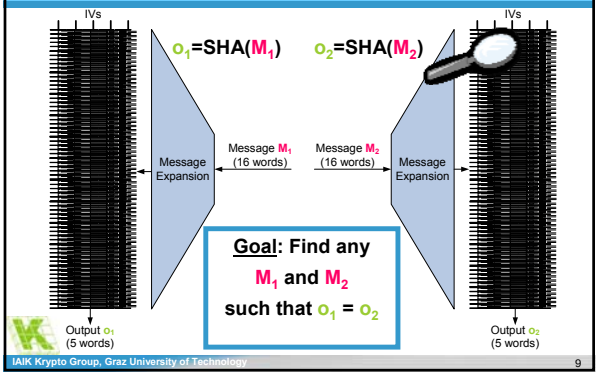
SHA-1

$$W_t = \begin{cases} M_t & for\,(0 \le t \le 15) \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) << 1 & for\,(16 \le t \le 79) \end{cases}$$

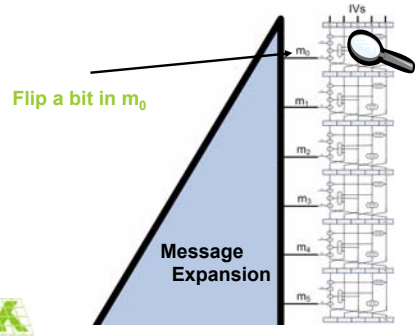
Outline of SHA – State Update



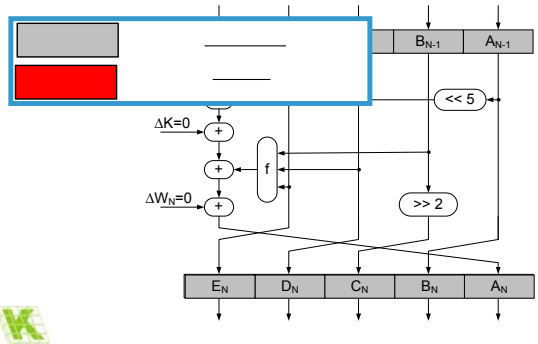
How to produce a collision?



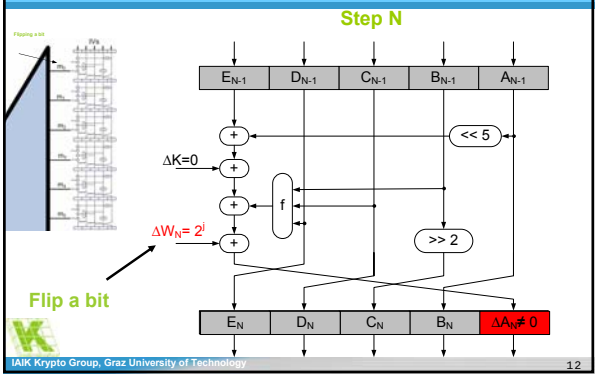
How to produce a collision: Chabaud-Joux



Effects of a flipping bit

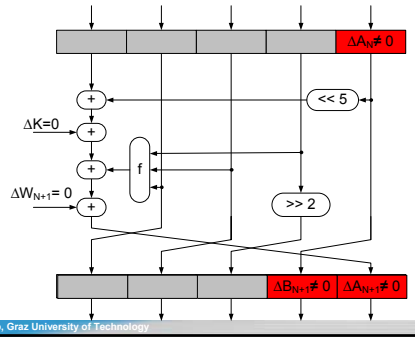


Effects of a flipping bit

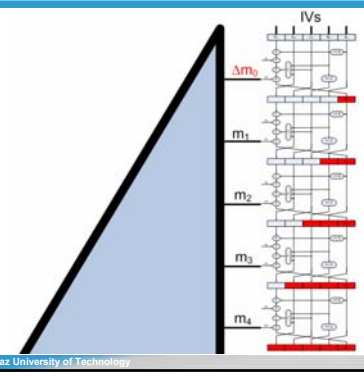


Effects of a flipping bit

Step N+1

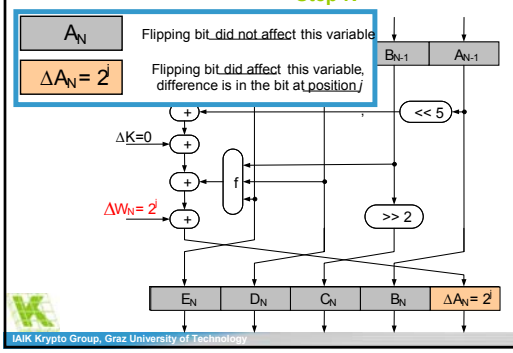


Effects of a flipping bit



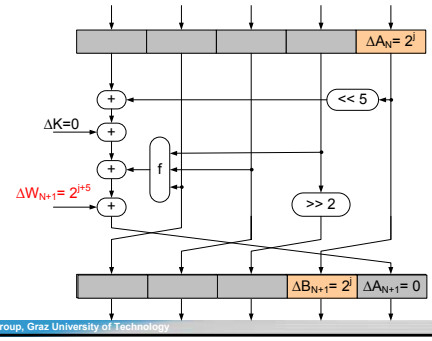
Producing a local collision

Step N



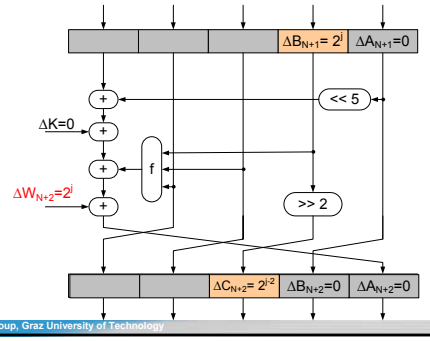
Producing a local collision

Step N+1



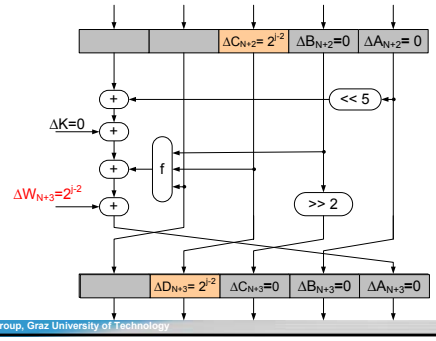
Producing a local collision

Step N+2



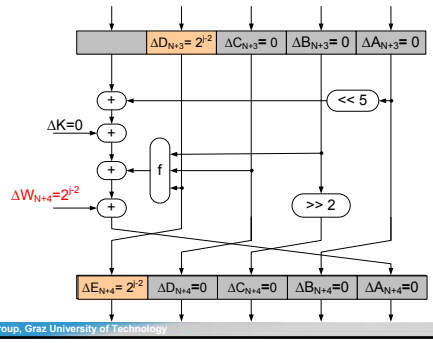
Producing a local collision

Step N+3



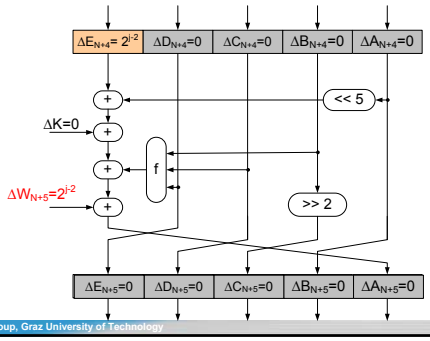
Producing a local collision

Step N+4

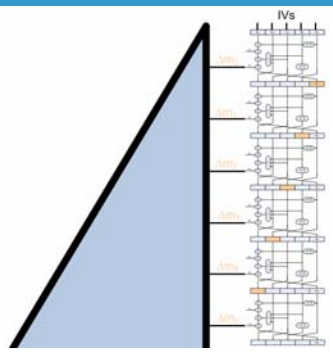


Producing a local collision

Step N+5



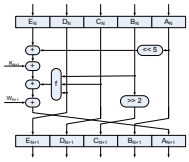
Producing a local collision



Forcing the Boolean functions to behave properly

Definition of rounds

Round i	Function f(i)	
	Name	Definition
0-19	IF	BC⊕ (~B)D
20-39	XOR	B⊕C⊕D
40-59	MAJ	BC⊕BD⊕CD
60-79	XOR	B⊕C⊕D



Truth table

BCD	XOR	IF
000	0	0
001	1	1
010	1	0
011	0	1
100	1	0
101	0	0
110	0	1
111	1	1

Conditions

Δ(BCD)	Δ(IF)	condition
000	0	always
001	1	b=0
010	1	b=1
011	0	never
100	1	c⊕d=1
101	0	b⊕c⊕d=0
110	0	b⊕c⊕d=1
111	1	c⊕d=0

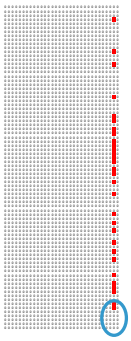
Forcing the modular addition: Z = X + Y

- We want that the carry bit propagation is not influenced by the difference
- Results in linear conditions between bits of input and output

From local collision to full collision

- Message expansion
  - Every bit changed influences other bits
  - Local collision producing differences have to be combined into admissible expanded word difference
- Linear expansion
  - Local collision producing differences have to be combined into admissible expanded word
  - Search for `small' collision-producing word

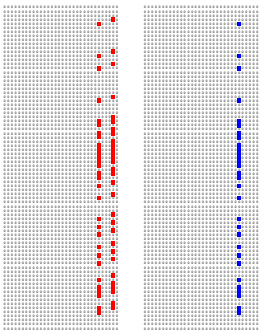
Building a collision-producing difference pattern



- Perturbation pattern
  - Low weight
  - Valid expanded message

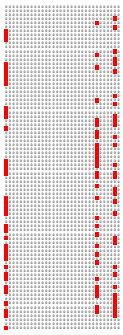
$$W_t = W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}$$

Building a collision-producing difference pattern



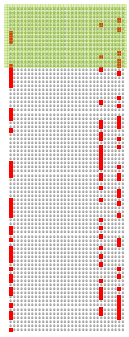
- Apply 5 corrections with the same pattern…

Building a collision-producing difference pattern



- Completed difference pattern consisting of
  - 1 perturbation pattern
  - 5 correction patterns

Building a collision-producing difference pattern



- Every perturbation gives us 2-5 conditions we need to fulfill during our final search
- Goal is to minimize these conditions

Application to SHA-1

- Perturbation patterns of weight 25 for SHA-0
  => less than 80 conditions => academic break

SHA-0

SHA-1

- For SHA-1, weight of patterns is bigger than 100
  ➔ much more than 80 conditions

Other view on the attack

1. Construct differential
   - Assume difference propagation as in linear approximation
     - Boolean functions → XOR
     - Additions → XOR
   - Find differential with low weight
2. Find pair of messages following the differential
   - First 15 rounds: direct computation
   - Next rounds: guess and verify
     - Some of the conditions can be translated into conditions on the message → easy to fulfil

Extensions and improvements

- Biham and Chen
  - Improve step 2 (+ apply to SHA-1)
- Rijmen and Oswald
  - Improve step 1 for SHA-1
- Wang et al.
  - Improve step 1 + apply to MD4, MD5, SHA-0, SHA-1
  - Improve step 2
- (Joux et al.)

Improve step 1: increased search space

- Perturbation-correction: all 6 words are valid expanded messages
  - 192 degrees of freedom for differential
- But: only the sum of the 6 words needs to be so
  - 352 degrees of freedom
- New approach to find differential
  - based on coding theory techniques

From SHA-1 to a linear code

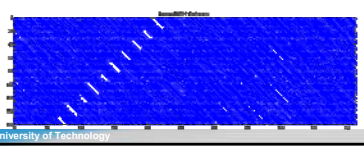
- Output of SHA-1 linearized

$$o = wA + b = mEA + b$$

- Collision if two messages have same hash value

$$m_1, m_2 = m_1 + \delta \quad \Rightarrow o_2 - o_1 = (m_1 + \delta)EA + b - (m_1 EA + b) = \delta EA = 0$$

linear code with check matrix $H_{160\times 512} = (EA)^T$ n = 512 k = 352



Using code theory

- Better: work with expanded message
  - codeword is valid expanded message
  - codeword is collision-producing
- Check matrix: 2208 x 2560
- Generator matrix: 352 x 2560

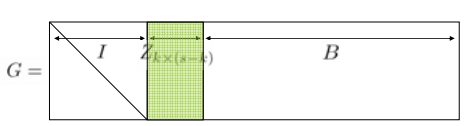
Léon's algorithm

- Randomly permute columns of G
- Perform Gaussian elimination

$$G = [I \; Z_{k\times(s-k)} \; B]$$

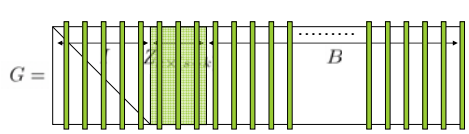
$$G^* = [I \; Z_{k\times(s-k)}]$$

- For all combinations of at most p rows of G^*
  - If wt (v*) ≤ p compute v and check weight of v



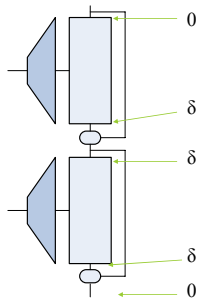
Optimizing Léon

- The code is not random
  - Symmetry: words can be rotated
  - Low-weight words: 1's occur in `bands'
- Force bits at positions 0,1,2,…u to zero



Improve step 1: Near-collisions

- Observation: near-collisions are much easier than collisions
- Two related near-collisions can give a collision in a multi-block approach

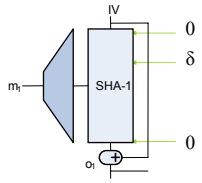


Improve step 2

- Pre-fulfilling conditions for more steps in the beginning (15 → 18, 19, 20 (?))
  - Neutral-bit technique [Biham,Chen]
  - Advanced message modification (?) [Wang et al.]

Improve step 1: Pseudo-collisions

- Observation: Pseudo-collisions are much easier than collisions
- Split differential in two parts:
  1. $0 \rightarrow \delta$: special
  2. $\delta \rightarrow 0$: pseudo-collision from linear approximation



Improve step 1: Pseudo-collisions

- Special part of the differential is constructed in different way
  - More freedom in search phase
  - Needs clever strategy in the beginning
  - Lower probability (= more conditions)
    - Not important because conditions here can be met easily
- Pseudo-collision over second part has much better probability

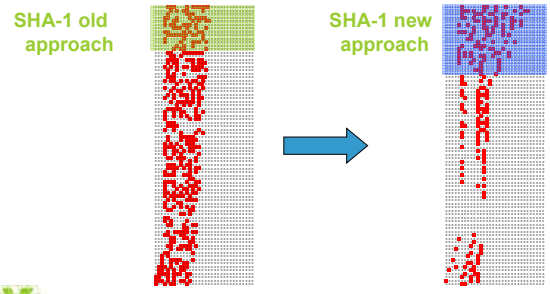
m_1

Overview

All methods can be described as the search for a low-weight word in a linear code

1. Perturbation-correction: $G_{192 \times 2560}$
2. General collision: $G_{352 \times 2560}$
3. Near-collision: $G_{704 \times 5120}$
   - + (2): $G_{864 \times 5120}$
4. Pseudo-collision: $G_{352 \times 2560}$
   - +(2),(3): $G_{672 \times 2560}$

Result for SHA-1



m_2

Concluding remarks

- MD4 structure has weaknesses
  - Remove them one by one?

  - Choose a completely different structure?
- Structures based on mixed use of XOR and modular addition need a large number of rounds