5: Dedicated Compression Functions
5.1: Number-Theory Based Compression Functions

- "Hard" problems, used in public-key crypto:
  - *Factorization Problem*: Given a composed number n, find any divisor $\mathbf{d}|n$, $\mathbf{d} \notin \{1, n\}$ (typically: $n = \mathbf{pq}$ the product of two huge random primes).
  - *RSA Problem*: Given $n = \mathbf{pq}$, as above, $e \equiv 1 \bmod \varphi(\mathbf{n})$ and $Z \in \mathbb{Z}_n^*$, find "an e-th root of $Z \bmod n$, i.e., a number $\mathbf{x}$ with $\mathbf{x}^e = Z \bmod n$.
  - *Discrete Logarithm Problem*: Given a prime p, a "generator" g, and a value $Z = g^{\mathbf{y}} \bmod p$, find $\mathbf{y}$.

Using “Hard” Problems

- Define HF, which is **provably** secure (collision resistant or whatever), assuming the given problem is actually hard.
- Proof works the other way: Assuming an attack against the hash function (e.g., an algorithm A to find a collision), show that you can then efficiently solve the “hard” problem, using A as a subroutine.
- Advantage for Hash-then-Sign: Possibly the same unproven assumption for both steps.
- Disadvantage: Lack of efficiency.

Using the RSA Problem

Let n and e be as above, and $c \in \mathbb{Z}_n^*$ random. Define the compression function

$$f : \{0,1\} \times \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_n^*, \quad f(a,x) = c^a * x^e \bmod n.$$

Theorem 7

If the RSA problem for (n,e) is "hard", then finding either collisions or second preimages or preimages for the compression function f is "hard".

Let a hash function H be defined by the Merkle-Damgård iteration of f, with MD strengthening.

Corollary 8

If the RSA problem for (n,e) is "hard", then finding either collisions or second preimages or preimages for the hash function H is "hard".

Chaum, van Heijst and Pfitzmann

A compression function, based on the discrete log problem

Let p and $q = (p-1)/2 \geq 3$ be primes and g a generator of $\mathbb{Z}_p$. Let B be a random number from $\mathbb{Z}_p$, and define the cvhp compression function:

$$\text{cvhp} : \mathbb{Z}_q \times \mathbb{Z}_q \longrightarrow \mathbb{Z}_p, \quad \text{cvhp}(x, y) = g^x * b^y \bmod p.$$

(Note that the the domain $\mathbb{Z}_q \times \mathbb{Z}_q$ is much larger than the range $\mathbb{Z}_p$, which makes cvhp a good compression function.

Theorem 9

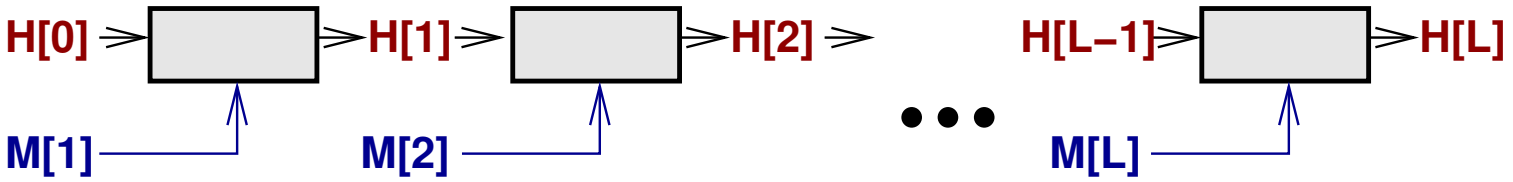
The cvhp function is collision resistant, if the Discrete Logarithm Problem in $\mathbb{Z}_p$ to the base g is "hard".

5.2: MD4 and Relatives

In the rest of this section:

- Straightforward “local” attacks
- basic structure for more complex attacks, extending the “local” attacks for the entire hash function
- Hash functions evolving from MD4 (MD5, SHA-0, SHA-1, RIPEMD, SHA-2)

5.3: The MD4 structure
Recall: Merkle-Damgård



- "chaining values" $H[i]$
- "initial value" $H[0]$.
- final hash: $H[L]$.
- message blocks $M[i]$.

MD4 Padding

- Compression function $\{0,1\}^{128} \times \{0,1\}^{512} \to \{0,1\}^{128}$, thus
  - "chaining values" 128 bit.
  - message blocks 512 bit.
- 10^*-padding:
  - always append a 1-bit
  - then append as many 0-bits as required, such that the message length is congruent to 448 mod 512.
  - This leaves 64 bit to encode the original message length (MD strengthening).

32-bit Words

- optimized for 32-bit machines
- low-level operations, efficient on 32-bit machines:
  - addition mod 2^{32} (**add**).
  - bitwise operations (**and**, **or**, **not**, **xor**, **rotate**).
- "initial value":

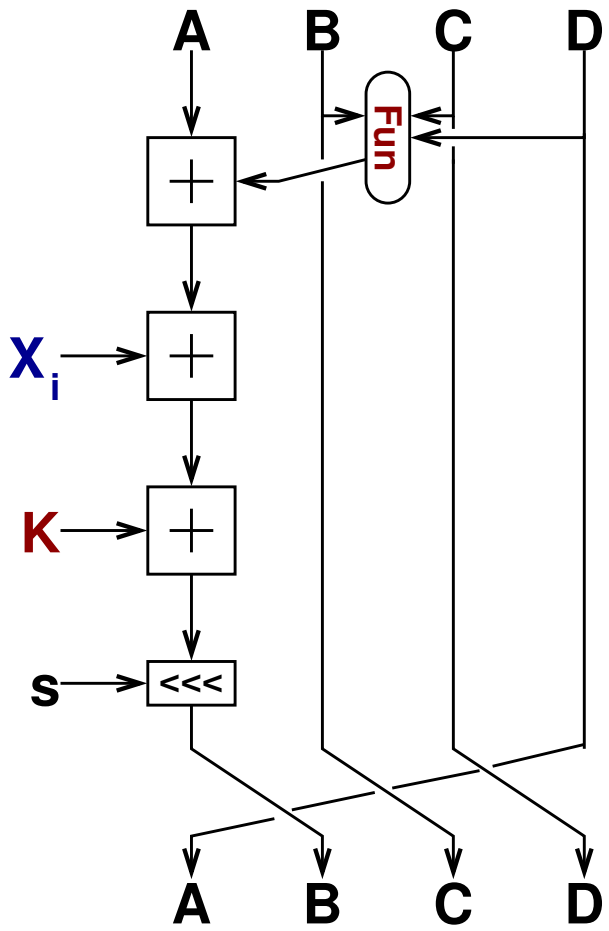
$$\begin{aligned} \mathbf{H[0]} &= (A, B, C, D) \\ &:= (\mathbf{01\,23\,45\,67},\ \mathbf{89\,ab\,cd\,ef},\\ &\quad\ \mathbf{fe\,dc\,ba\,98},\ \mathbf{76\,54\,32\,10}) \end{aligned}$$

The MD4 Compression Function (3-round block cipher, Davies-Meyer)

- Input:
  - $H[i-1] = (A, B, C, D) \in \{0,1\}^{128}$ and
  - $M[i] = (X_0, X_1, \ldots, X_{15}) \in \{0,1\}^{512}$;
- $(AA, BB, CC, DD) := (A, B, C, D)$;
- $(A, B, C, D) := Round_1((A, B, C, D),\ (X_0, X_1, \ldots, X_{15}))$;
- $(A, B, C, D) := Round_2((A, B, C, D),\ (X_0, X_1, \ldots, X_{15}))$;
- $(A, B, C, D) := Round_3((A, B, C, D),\ (X_0, X_1, \ldots, X_{15}))$;
- $A := A + AA$; $B := B + BB$; $C := C + CC$; $D := D + DD$; (**add** mod 2^{32}).
- Output: $H[i] = (A, B, C, D)$.

5.4: MD4 Details

- message split into 16 words $X_0, \ldots, X_{15}$
- each round: 16 steps ($\rightarrow$ Figure)
- each step:
- inject $\mathbf{X_i}$, **add**, **rotate** ("<<<")
- round-dependent
  - constant **K**,
  - nonlinear bitwise **Fun**,
- step- and round-dependent: <<< **s**
- Assuming **Fun** and **K** being given, we define a procedure Step(**i**, **s**).

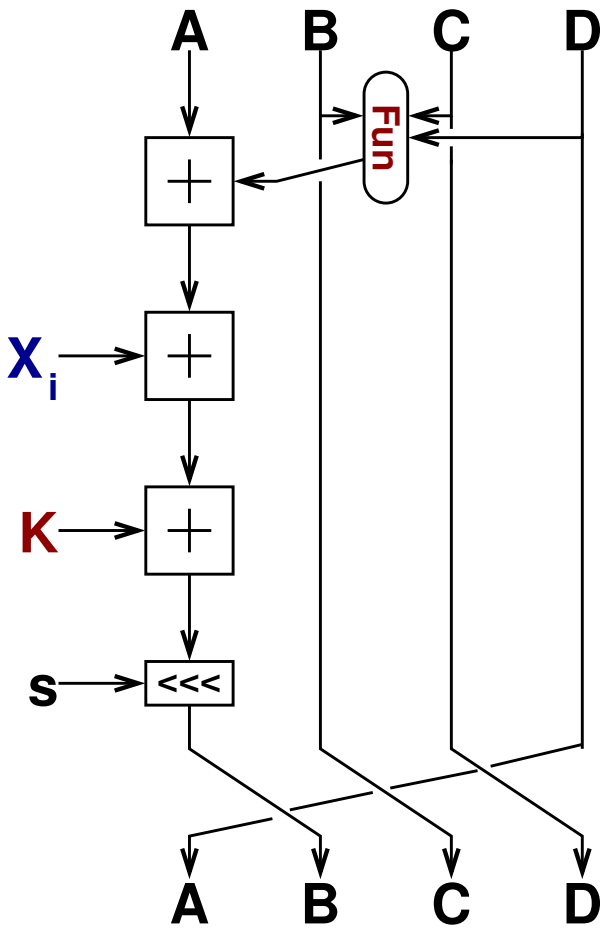


Observation

- each step changes only one 32-bit word of the state
- the others are just moved
- formula for changing the word:

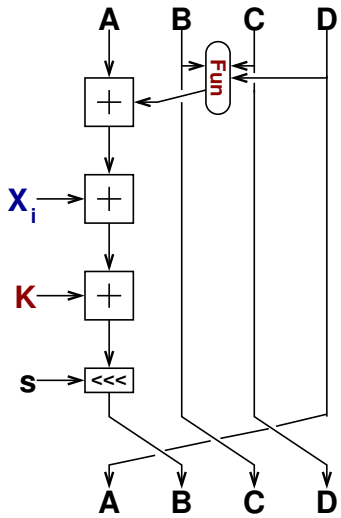
$$(\mathbf{A} + \mathbf{Fun}(\mathbf{B}, \mathbf{C}, \mathbf{D}) + \mathbf{X_i} + \mathbf{K}) <<< \mathbf{s}$$

- Step(i, s) = Use message word X_i and rotation constant s.



Round 1

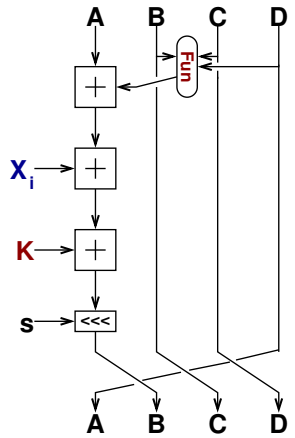
Fun =**F**; ($*$ if-then-else $*$):
F(X,Y,Z) = (X **and** Y) **or** (**not**(X) **and** Z).
K $= 0$.



Step(0, 3)	Step(1, 7)	Step(2,11)	Step(3,19)
Step(4, 3)	Step(5, 7)	Step(6,11)	Step(7,19)
Step(8, 3)	Step(9, 7)	Step(10,11)	Step(11,19)
Step(12, 3)	Step(13, 7)	Step(14,11)	Step(15,19)

Round 2

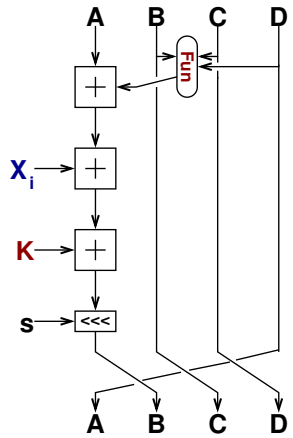
Fun =**G**; (∗ majority ∗):
G(X,Y,Z) = (X **and** Y) **or** (X **and** Z) **or** (Y **and** Z).
K = 5*a*827999.



Step(0, 3)	Step(4, 5)	Step(8, 9)	Step(12, 13)
Step(1, 3)	Step(5, 5)	Step(9, 9)	Step(13, 13)
Step(2, 3)	Step(6, 5)	Step(10, 9)	Step(14, 13)
Step(3, 3)	Step(7, 5)	Step(11, 9)	Step(15, 13)

Round 3

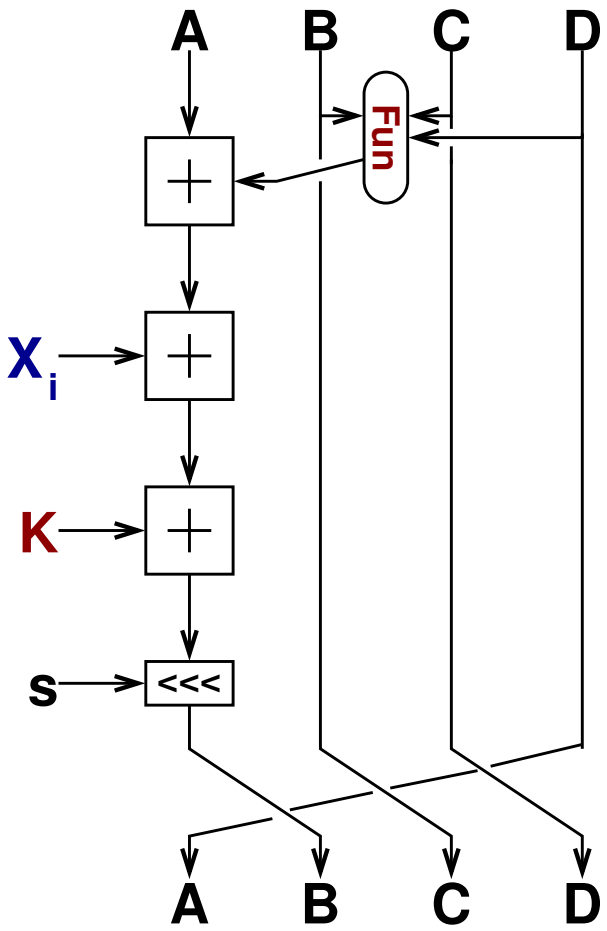
Fun =**H**; (∗ parity ∗):
H(X,Y,Z) = X **xor** Y **xor** Z.
K = 6*ed*9*eba*1.



Step(0, 3)	Step(8, 9)	Step(4, 11)	Step(12, 15)
Step(2, 3)	Step(10, 9)	Step(6, 11)	Step(14, 15)
Step(1, 3)	Step(9, 9)	Step(5, 11)	Step(13, 15)
Step(3, 3)	Step(11, 9)	Step(7, 11)	Step(15, 15)

5.5: Initial Cryptanalysis

- consider one single round of MD4
- it is easy to find collisions – and even preimages and second preimages.
- **Preimage? How?**



The effect of flipping one bit:

Consider random 32-bit values X, Y and Z and any index $t \in \{1 \ldots 32\}$ and flipping the t-th bit in either X, Y or Z (i.e., XORing by $(0^{32-t}, 1, 0^{t-1})$). The output of **Fun**(X, Y, Z) will be affected as follows:

1. $U =$ **F**(X,Y,Z) = (X **and** Y) **or** (**not**(X) **and** Z):
   - With prob. 1/2: U does not change.
   - With prob. 1/2: $U := U \oplus (0^{32-t}, 1, 0^{t-1})$ ("the t-th bit changes")
2. Same for $V =$ **G**(X,Y,Z) = (X **and** Y) **or** (X **and** Z) **or** (Y **and** Y).
3. $W =$ **H**(X,Y,Z) = X **xor** Y **xor** Z:
   - With prob. 1: $W := W \oplus (0^{32-t}, 1, 0^{t-1})$

The effect of flipping two bits:

Consider the setting as above, but flip the t-th bit in **two** of the three inputs X, Y, and Z. How does this affect the output of F, G, and H?

Cryptanalysis

- Initial value (A, B, C, D).
- Search for two different messages $X = (X_0, X_1, \ldots, X_{15})$ and $X' = (X'_0, X'_1, \ldots, X'_{15})$ with Hash(X) = Hash(X') for
- Hash = $\langle$ *many steps of MD4* $\rangle$.

The two-step approach

1. Search for a "good" difference δX between X and X':
   - This determines not only the final difference of zero, but also "expected" differences $\delta_i = X_i \oplus X'_i$ after each step $i \in \{1, 2, \ldots\}$.
2. Find X_0 and X_0 with Hash(X_0) = Hash(X'_0):
   - Start with a random X_0. This determines X'_0.
   - **While** Hash(X_0) $\neq$ Hash(X'_0) **loop**
     - Search for the first step i' with an "unexpected" difference $X_i \oplus X'_i \neq \delta_i$.
     - Change X_{i-1} (and thus X'_{i-1}) to ensure the expected difference δ_i.
     - Go backward to compute X_0 and X'_0.

Examples ($\rightarrow$ Blackboard)

- $X_i = X'_i$, for $i \notin \{7, 11\}$; first 29 steps
- $X_i = X'_i$, for $i \notin \{0, 1\}$; steps 3–32
- $X_i = X'_i$, for $i \notin \{1, 2, 12\}$; [Wang et al.] with

$$\begin{aligned} X'_1 - X_1 &= 2^{31} \\ X'_2 - X_2 &= 2^{31} - 2^{28} \\ X'_{12} - X_{12} &= 2^{31} \end{aligned}$$

  “Local” collision for steps 36–41.

MD4 Overview

16 steps:

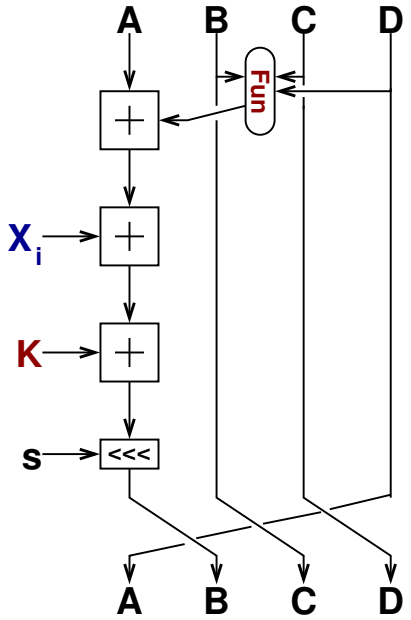
Step(0, 3)	Step(1, 7)	Step(2,11)	Step(3,19)
Step(4, 3)	Step(5, 7)	Step(6,11)	Step(7,19)
Step(8, 3)	Step(9, 7)	Step(10,11)	Step(11,19)
Step(12, 3)	Step(13, 7)	Step(14,11)	Step(15,19)

second round:

Step(0, 3)	Step(4, 5)	Step(8, 9)	Step(12,13)
Step(1, 3)	Step(5, 5)	Step(9, 9)	Step(13,13)
Step(2, 3)	Step(6, 5)	Step(10, 9)	Step(14,13)
Step(3, 3)	Step(7, 5)	Step(11, 9)	Step(15,13)

3rd round:

Step(0, 3)	Step(8, 9)	Step(4,11)	Step(12,15)
Step(2, 3)	Step(10, 9)	Step(6,11)	Step(14,15)
Step(1, 3)	Step(9, 9)	Step(5,11)	Step(13,15)
Step(3, 3)	Step(11, 9)	Step(7,11)	Step(15,15)



5.6: Standardized Offsprings of MD4

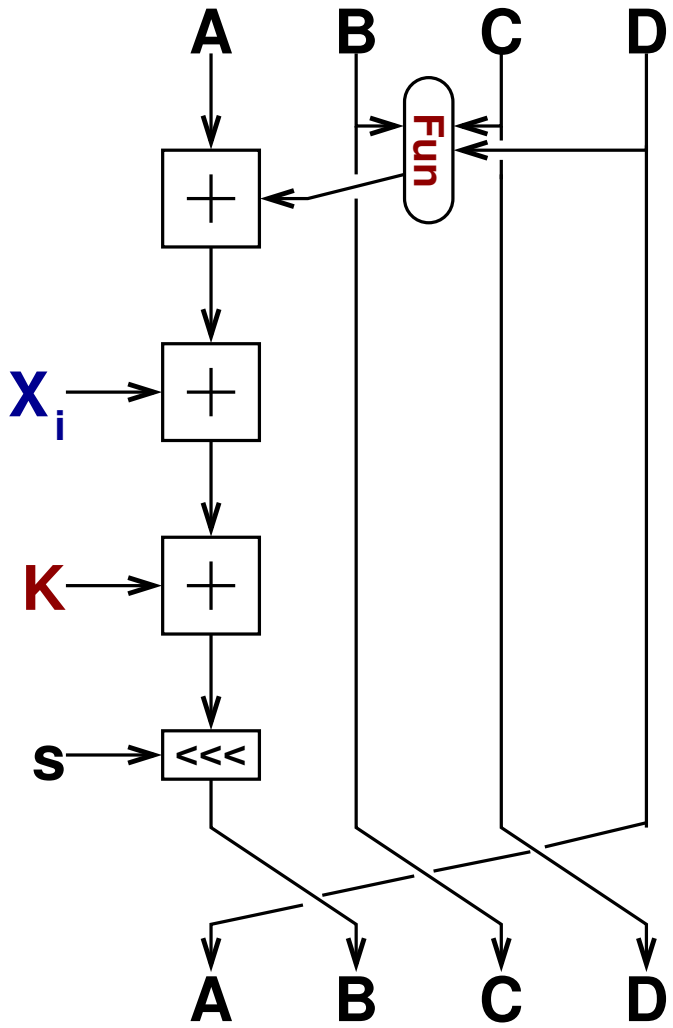
- Early doubts about the security of MD4.
- No practical relevance for MD4 itself.
- But inspiration for many other hash functions with great practical relevance:

1991:	MD5 (Rivest)
1992:	RIPE-MD (European research project RIPE)
1993:	SHA(-0) (NSA)
1995:	SHA-1 (NSA)
	RIPEMD-160 (Bosselaers, Dobbertin, Preneel)
2001	SHA-2 Family

- Below, we will describe the major modifications/extensions.

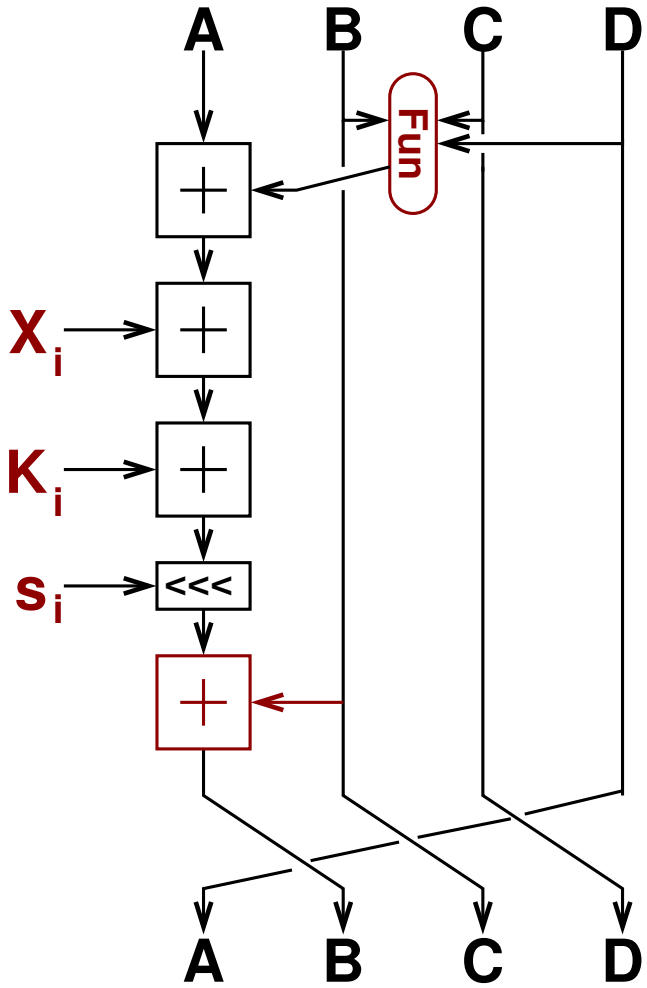
Recall the MD4 structure

- 16 steps per round
- 1 step per message word X_i,
- **add** and **rotate** (“<<<”)
- round-dependent:
  - const **K**,
  - nonlinear function **Fun**,
- step-dependent:
  - rotation count **s**
- For given **Fun** and **K** we define Step(**i**, **s**).



MD5

- 4 rounds (instead of 3)
- functions **Fun**:
  changed (second round),
  new one (4th round)
- rounds 2–4: more complex order of message words X_i
- step-dependent constant K_i (instead of round-dependent constant)
- optimized rotation-counts s_i
- additional Addition



Collisions for the MD5 Compression Function

1994 Den Boer, Bosselars ("Pseudo-Collision"):
Find $\mathbf{H_0}, \neq \mathbf{H'_0}$ and $\mathbf{X}$ with

$$\text{MD5-Compress}(\mathbf{H_0}, \mathbf{X}) = \text{MD5-Compress}(\mathbf{H'_0}, \mathbf{X})$$

1996 Dobbertin: Find $\mathbf{H_0}$ and $\mathbf{X} \neq \mathbf{Y}$ with

$$\text{MD5-Compress}(\mathbf{H_0}, \mathbf{X}) = \text{MD5-Compress}(\mathbf{H'_0}, \mathbf{Y})$$

Two reasons that should have been sufficient to abandon MD5.

These results have been completely ignored by the industry.

The lousy phrasing ("Pseudo-Collision") has not been helpful.

Collisions for the MD5 Hash Function

2004 Wang et al.
Given $\mathbf{H_0}$ find $(\mathbf{X_1}, \mathbf{X_2}) \neq (\mathbf{Y_1}, \mathbf{Y_2})$with

$$\begin{aligned} & \text{MD5-Compress}(\ \text{MD5-Compress}(\mathbf{H_0}, \mathbf{X_1}),\ \mathbf{X_2}\ ) \\ = \quad & \text{MD5-Compress}(\ \text{MD5-Compress}(\mathbf{H_0}, \mathbf{Y_1}),\ \mathbf{Y_2}\ ) \end{aligned}$$

At the compression function level:

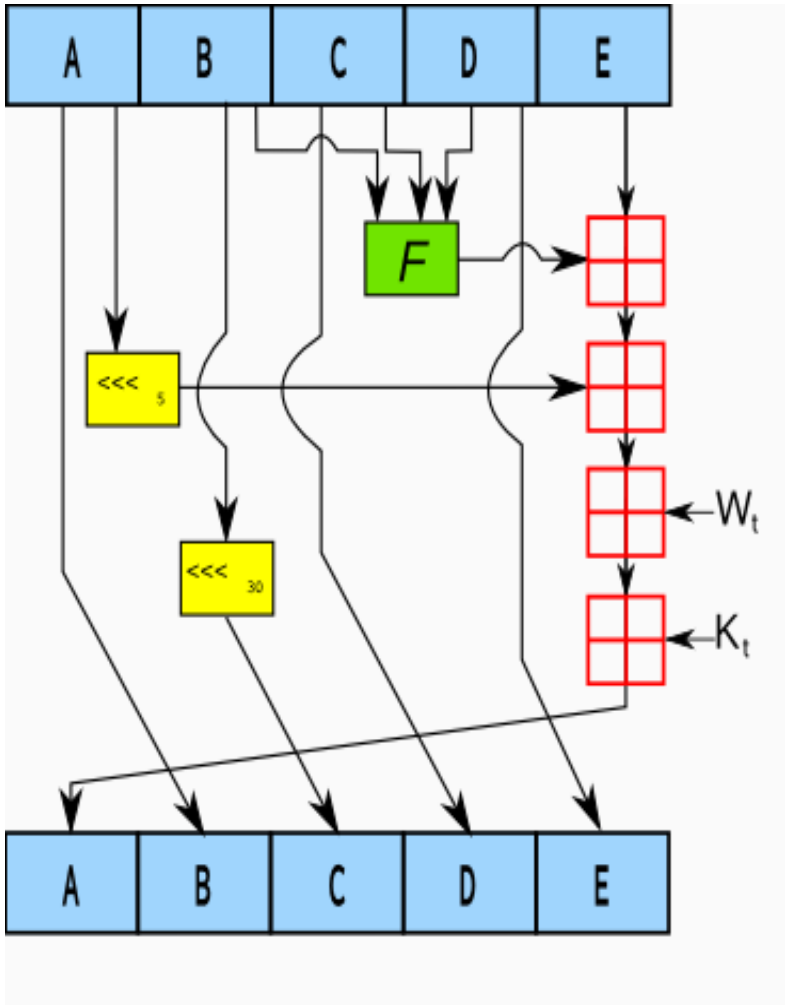
Near-Collision + Pseudo-Collision

At this point of time, MD5 has been abandoned almost in panic.

SHA-0, SHA-1

SHA= “Secure Hash Algorithm”

- 160 bit result (5 32-bit words)
- novel structure ($\rightarrow$ fig.)
- 5 rounds (80 steps)
- “message schedule”



The message schedule (given a message block $w[0], \ldots, w[15]$)**:**

SHA-0 $w[i] := w[i-3] \oplus w[i-8] \oplus w[i-14] \oplus w[i-16]$

SHA-1 $w[i] := (w[i-3] \oplus w[i-8] \oplus w[i-14] \oplus w[i-16]) \lll 1$

Evolution

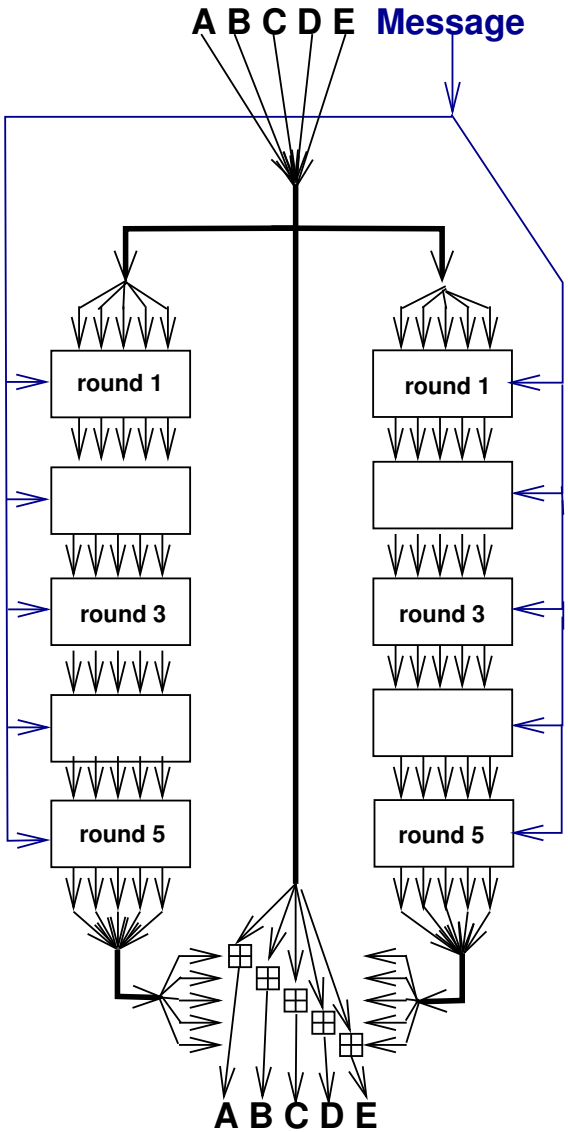
1993: standard “Secure Hash Algorithm” (SHA)

1995: SHA-1 replaces SHA (now SHA-0)
single difference: rotation in message schedule
no reasons given by the authors

Collisions:

	# Steps	workload	publication
SHA-0	80 (full)	2^{61}	1998
	80 (full)	2^{51}	2004
	80 (full)	2^{39}	2005
SHA-1	53	$< 2^{80}$	Jan. 2005
	80 (full)	2^{69}	Feb. 2005
	80 (full)	2^{61}	Aug. 2005
	64	2^{35}	Dec. 2006

RIPE-MD and RIPE-MD-160



- 2 parallel strands, each like MD4
- each strand with different constants

1992 RIPE-MD (3 rounds, 128 bit)

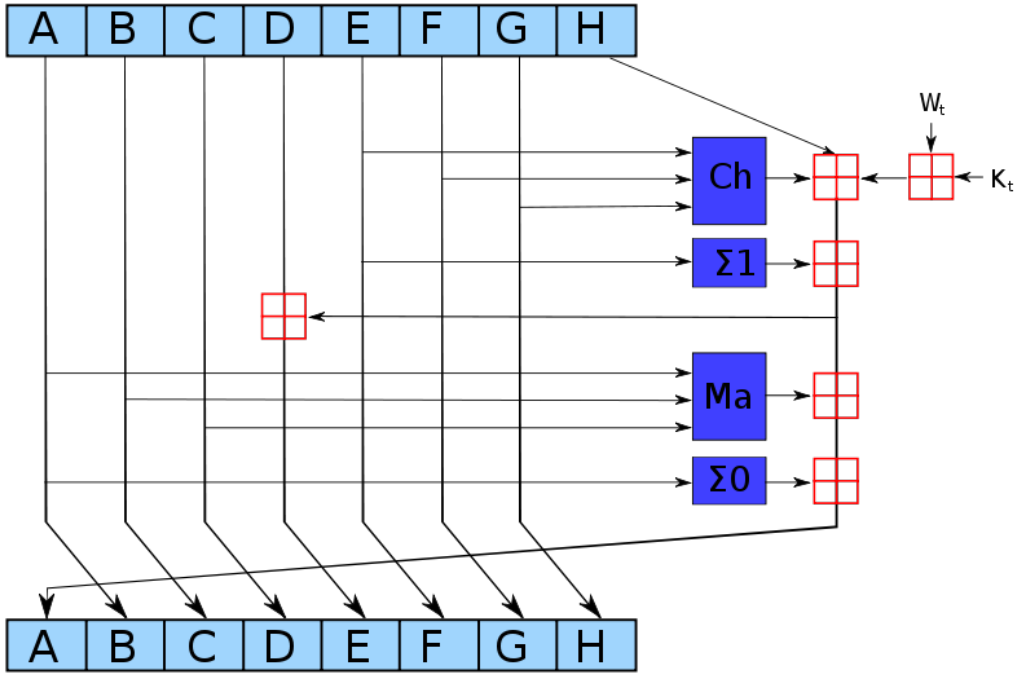
1996 RIPE-MD-160 (5 rounds, 160 bit)

2004 Collisions for RIPE-MD

2012 still no collisions for RIPE-MD-160 (faster than the generic 2^{80})

SHA-2

- SHA-256: 8 32-bit words, 64 steps
- SHA-512: 8 64-bit words, 80 steps
- as of April 2012: no attacks close to the full hash or compression function



- reduced output size: SHA-224, SHA-384, SHA-512/t (different IVs, truncated results)
- "choose": $\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$
- "majority" $\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$
- $\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$
  $\Sigma_1(E) = (E \ggg 6) \oplus (A \ggg 11) \oplus (A \ggg 25)$
  (for SHA-256; SHA-512: other rotation constants)