



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY

# Computer Security and Cryptography

CS381

来学嘉

计算机科学与工程系 电院3-423室

34205440 1356 4100825 laix@sjtu.edu.cn

2015-05



# Organization

- Week 1 to week 16 (2015-03 to 2014-06)
- 东中院-3-102
- Monday 3-4节; week 9-16
- Wednesday 3-4节; week 1-16
- lecture 10 + exercise 40 + **random tests** 40 + other 10
- Ask questions in class – counted as points
- Turn ON your mobile phone (after lecture)
- Slides and papers:
  - <http://202.120.38.185/CS381>
    - computer-security
  - <http://202.120.38.185/references>
- TA: Geshi Huang [gracehgs@mail.sjtu.edu.cn](mailto:gracehgs@mail.sjtu.edu.cn)
- Send homework to the TA

**Rule: do the homework on your own!**



# Contents

- Introduction -- What is security?
- Cryptography
  - Classical ciphers
  - Today's ciphers
  - Public-key cryptography
  - Hash functions and MAC
  - Authentication protocols
- Applications
  - Digital certificates
  - Secure email
  - Internet security, e-banking
- Computer and network security
  - Access control
  - Malware
  - Firewall



# Content



- Hash function – usage and basic properties
- Iterated hash function – Relationship between Hash function and its round (compress) function
- Real **compress functions**
  - Using block cipher
  - Dedicated hash functions, MD5,SHA1
- Security and attacks
- SHA-3
- MAC



# References

- Bart Preneel, The State of Cryptographic Hash Functions,  
<http://www.cosic.esat.kuleuven.ac.be/publications/>
- G. Yuval, "How to swindle Rabin," *Cryptologia*, Vol. 3, 1979, pp. 187-189
- Ralph Merkle. One way Hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology: CRYPTO 89*, LNCS 435. Springer-Verlag. 1989: 428–446.
- Ivan Damgård. A design principle for Hash functions. In Gilles Brassard, editor, *Advances in Cryptology: CRYPTO 89*, LNCS 435. Springer-Verlag. 1989:416~427.
- ISO/IEC 10118, Information technology - Security techniques - Hash-functions,
  - Part 1: General",
  - Part 2: Hash-functions using an n-bit block cipher algorithm,"
  - Part 3: Dedicated hash-functions,"
  - Part 4: Hash-functions using modular arithmetic,"
- M. Naor, M. Yung, "Universal one-way hash functions and their cryptographic applications," Proc. 21st ACM Symposium on the Theory of Computing, 1990, pp. 387-394.
- X. Lai, J.L. Massey, "Hash functions based on block ciphers," *Advances in Cryptology, Proceedings Eurocrypt'92*, LNCS 658, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 55-70
- L.R. Knudsen, X. Lai, B. Preneel, "Attacks on fast double block length hash functions," *Journal of Cryptology*, Vol. 11, No. 1, Winter 1998, pp. 59-72.



# References

- Joux, "Multicollisions in Iterated Hash Functions. Applications to Cascaded Constructions," *Crypto 2004 Proceedings*, Springer-Verlag, 2004.
- John Kelsey and Bruce Schneier Second Preimages on n-bit Hash Functions for Much Less than  $2^n$  Work, Eurocrypt 2005,
- Ronald Rivest. The MD4 Message Digest Algorithm. RFC1320, <http://rfc.net/rfc1320.html>. April 1992.
- Ronald Rivest. The MD5 Message Digest Algorithm. RFC1321, <http://rfc.net/rfc1321.html>. April 1992.
- Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In Dieter Gollmann, editor, *Fast Software Encryption*, Cambridge, UK, Proceedings, LNCS-1039. Springer.1996: 71~82.
- NIST. Secure Hash standard. Federal Information Processing Standard. FIPS-180-1. April 1995
- Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. *Cryptology ePrint Archive*, Report 2004/199, 2004. <http://eprint.iacr.org/2004/199.pdf>
- Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Crypt-analysis of the Hash Functions MD4 and RIPEMD, *Advances in Cryptology – EUROCRYPT 2005*, LNCS-3494. Springer.2005: 1~18..
- NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition". [NIST](#). 2012-10-02.
- G Bertoni,et al, Sponge functions, ECRYPT hash workshop, 2007
- **Draft FIPS 202, SHA-3 Standard**



# Constructions of compress functions

- Hash function **based on block ciphers**
  - Single length, double length
- **Dedicated** hash functions
  - MD2, MD4, MD5
  - SHA-0,SHA-1,SHA-256,SHA-384,SHA-512
  - RipeMD,RipeMD-128,RipeMD-160
  - HAVAL
  - Tiger, Whirlpool
- Hash functions using **modular operations**



# Hash functions in Standards

ISO 10118 (4 parts)

- Part 1: General (structure, padding, parameters)
- Part 2: block cipher based
- Part 3: dedicated hash functions (SHA-1, SHA-2, RIPEMD-128, RIPEMD-160, Whirlpool)
- Part 4: using modular operation

NIST FIPS PUB 180

- 180 (1993): secure hash algorithm, (SHA-0)
- 180-1 (1995) SHA-1 (critical modification)
- 180-2 (2002) SHA-2 (224, 256, 384, 512)

IETF RFC 1321, MD5



# Hash-functions from block ciphers

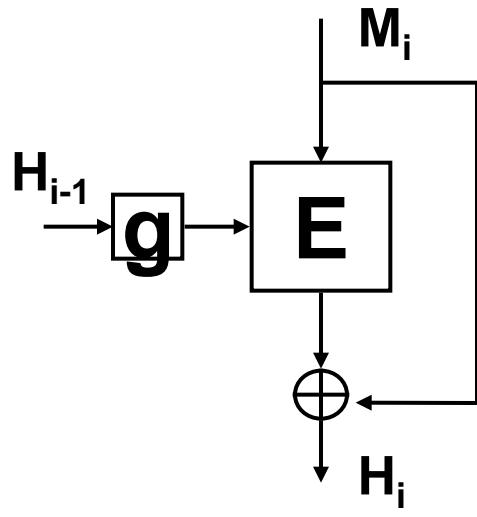


- ISO/IEC 10118-2
- Obtain a hash-function from an  $m$ -bit block cipher.
  - *Method 1* - hash-codes up to  $m$  bits long,
  - *Methods 2 & 3* - hash-codes up to  $2m$  bits,
  - *Method 4* - hash-codes up to  $3m$  bits long.
- Basic method: **Davies-Meyer** construction.
  - one-way function from a permutation:

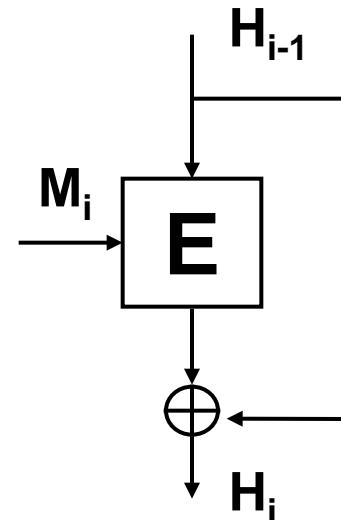
$$h(x,k) = e_k(x) \oplus x$$



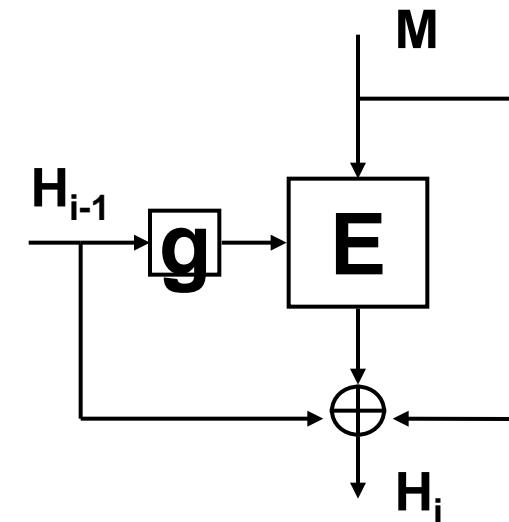
# Single length



Matyas-Meyer-Oseas



Davies-Meyer



Miyaguchi-Preneel

More details, double-length constructions, etc.  
see Ref. ISO-10118, works of Preneel



# Dedicated Hash functions



Specifically designed hash functions

- MD2, MD4, **MD5**
- HAVAL
- RipeMD, **RipeMD-128**, **RipeMD-160**
- SHA-0, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**
- Compress (round) function  $h$  using basic operations on blocks of 32/64 bits: XOR, AND, add, rotation, shift,...
- $h$  contains **i rounds  $\times$  j steps**, each step uses a non-linear function, and they are the same in each round.
- $h$  can be considered as a Davies-Meyer construction with a specially designed block cipher.
- More efficient:  $h$  can process more bits with fewer operations.



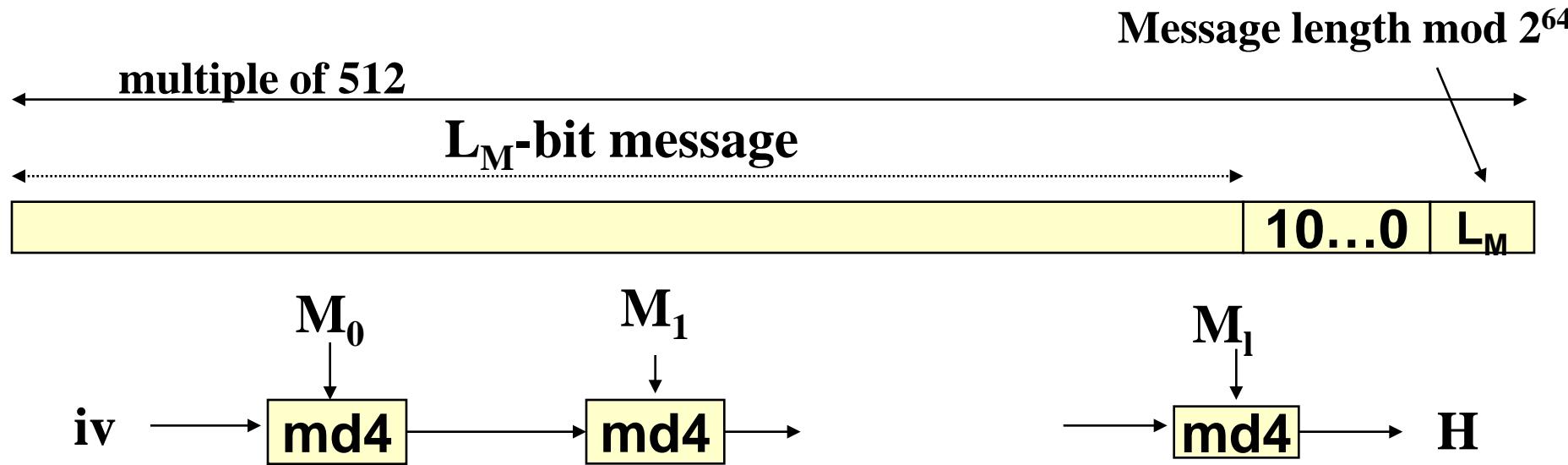
# MD4



- designed by Rivest in 1990, **128 bit** output
- for software implementation on 32-bit machines
- define  $f,g,h$  non-linear auxiliary function
- process 16-word (**512-bit**) message blocks in **3 rounds** ( $f,g,h$ )
- Each round has **16 step** operations on message subblocks and chaining value
- starting base for MD5, SHA and RIPEMD
- IETF RFC 1320



# MD4 Padding rule



- Padding: add a  $1, 00..0$  until last block has 512-64 bits.
- bit-byte-word as integer:
  - In byte: most significant bit first
  - In word: least byte first.



# MD4



- Initialize Message Digest Buffer:
  - 4 Word Buffer (A, B, C, D), each 32 Bit  
Word A: 01 23 45 67  
Word B: 89 ab cd ef  
Word C: fe dc ba 98  
Word D: 76 54 32 10
- 3 auxiliary functions: (X,Y,Z are 32-bit words)
  - $f(X, Y, Z) = XY \vee \text{not}(X)Z$  (ch )
  - $g(X, Y, Z) = XY \vee XZ \vee YZ$  (Majority)
  - $h(X, Y, Z) = X \oplus Y \oplus Z$  (parity)
- + denotes addition mod  $2^{32}$

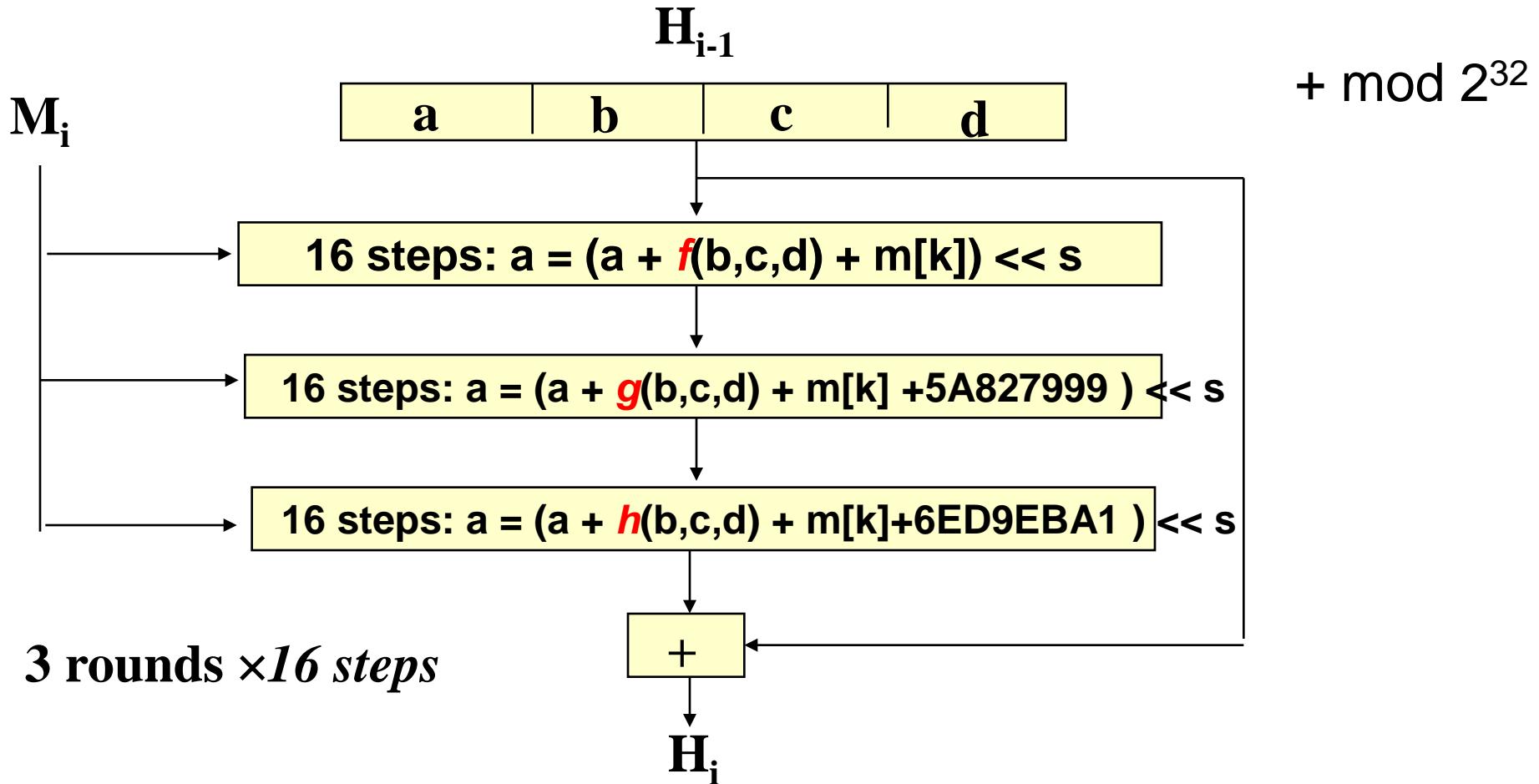


# MD4 compress function

- 512-bit message block  $m_i = (m[0], m[1], \dots, m[15])$
- then three 16-step rounds  $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$
- Round 1: for  $j=1$  to  $15$ , //  $s(j)=(3,7,11,19,3,7,11,19,\dots)$ 
  - $A \leftarrow (A + f(B, C, D) + m[j]) \ll s(j)$ ,
  - $(A, B, C, D) \leftarrow (D, A, B, C)$
- Round 2: for  $j=0$  to  $15$ , //  $s(j)=(3,5,9,13,3,5,9,13,\dots)$  (step 16-31)
  - $A \leftarrow (A + g(B, C, D) + m[j] + 5A827999) \ll s(j)$ ,
  - $(A, B, C, D) \leftarrow (D, A, B, C)$
- Round 3: for  $j=0$  to  $15$ , //  $s(j)=(3,9,11,15,3,9,11,15,\dots)$  step32-47
  - $A \leftarrow (A + h(B, C, D) + m[j] + 6ED9EBA1) \ll s(j)$ ,
  - $(A, B, C, D) \leftarrow (D, A, B, C)$
- $(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$
- $5A827999$  is  $2^{1/2}$ ,  $6ED9EBA1$  is  $3^{1/2}$



# MD4 compress function



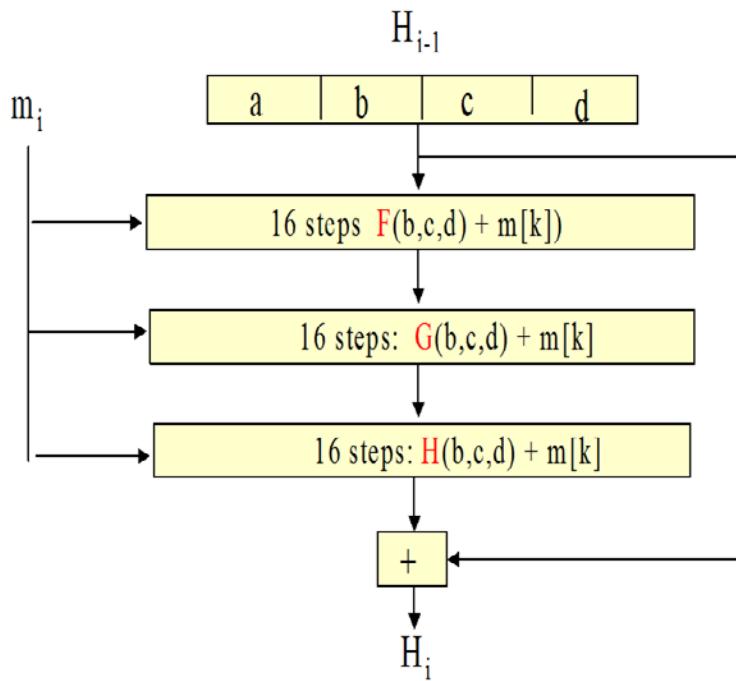
- 512-bit message block  $M_i = (m[0], m[1], \dots, m[15])$



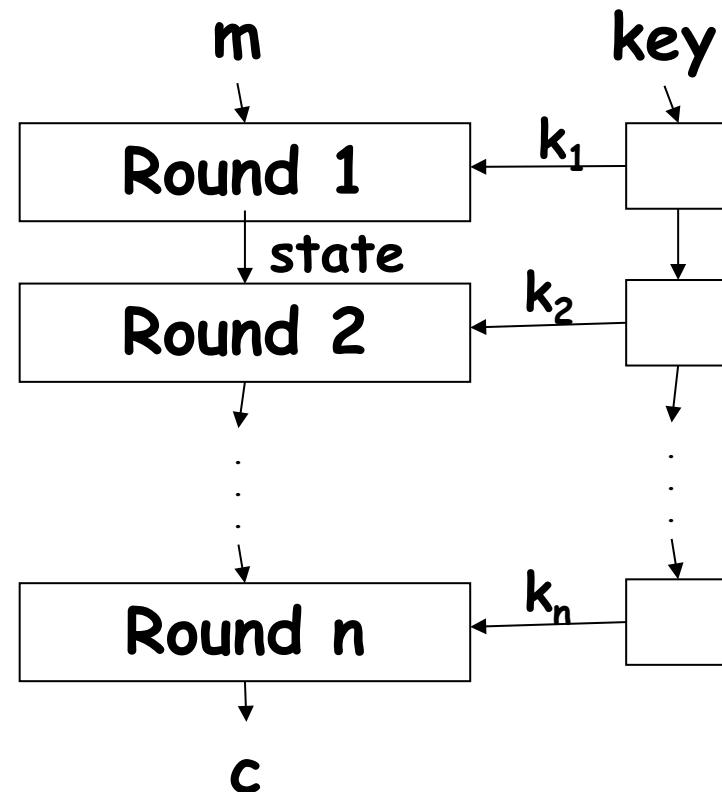
# Hash-step vs cipher-round



md4

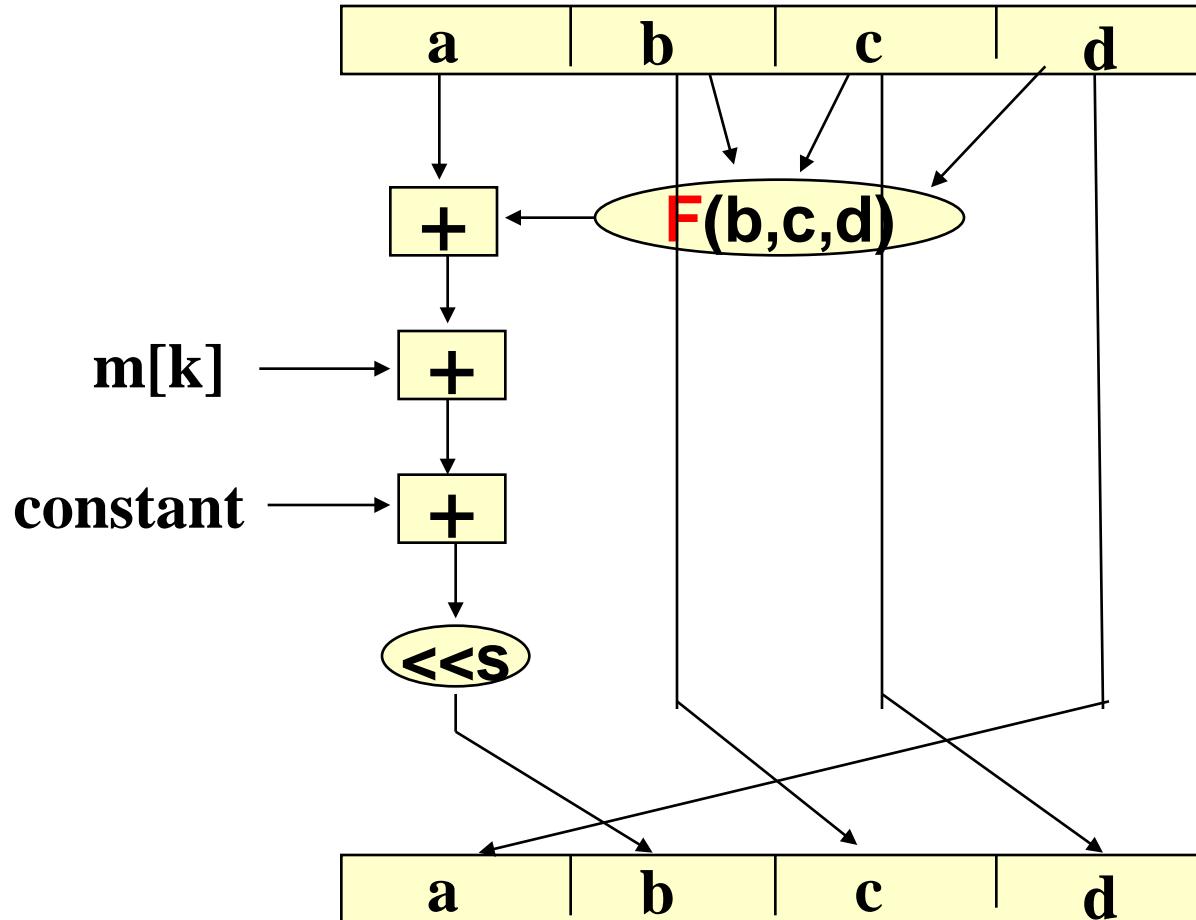


cipher





# MD4 step function



4-block  
feistel structure

+ mod  $2^{32}$



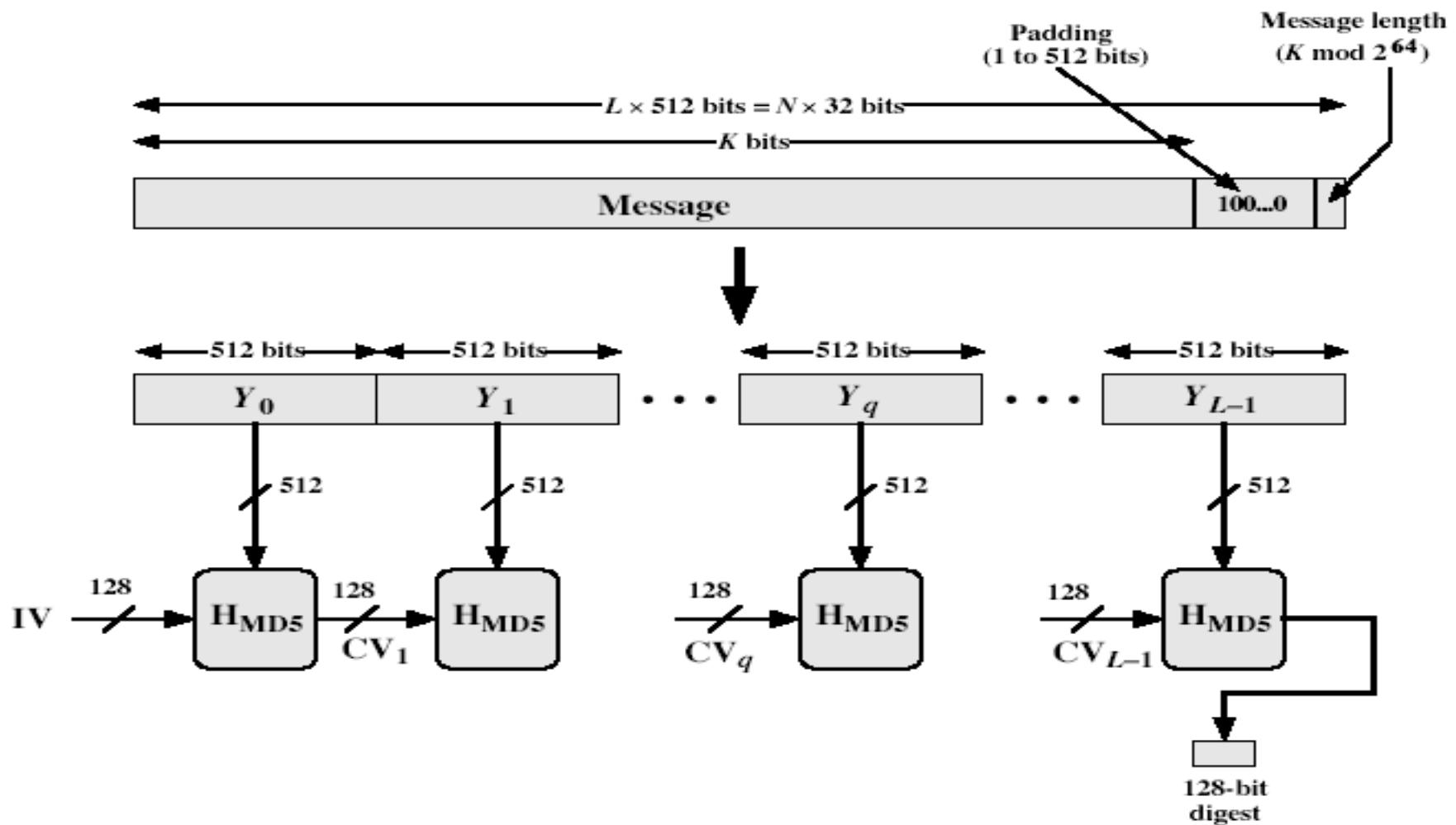
# MD5



- Designed by Rivest in 1992 as improvement of MD4
- Use 4 auxiliary functions:  $f,g,h,i$
- process 16-word (**512-bit**) message blocks in **4 rounds** ( $f,g,h,i$ )
- Each round has **16 step** operations on message subblocks and chaining value
- **128** bit output
- IETF RFC 1321

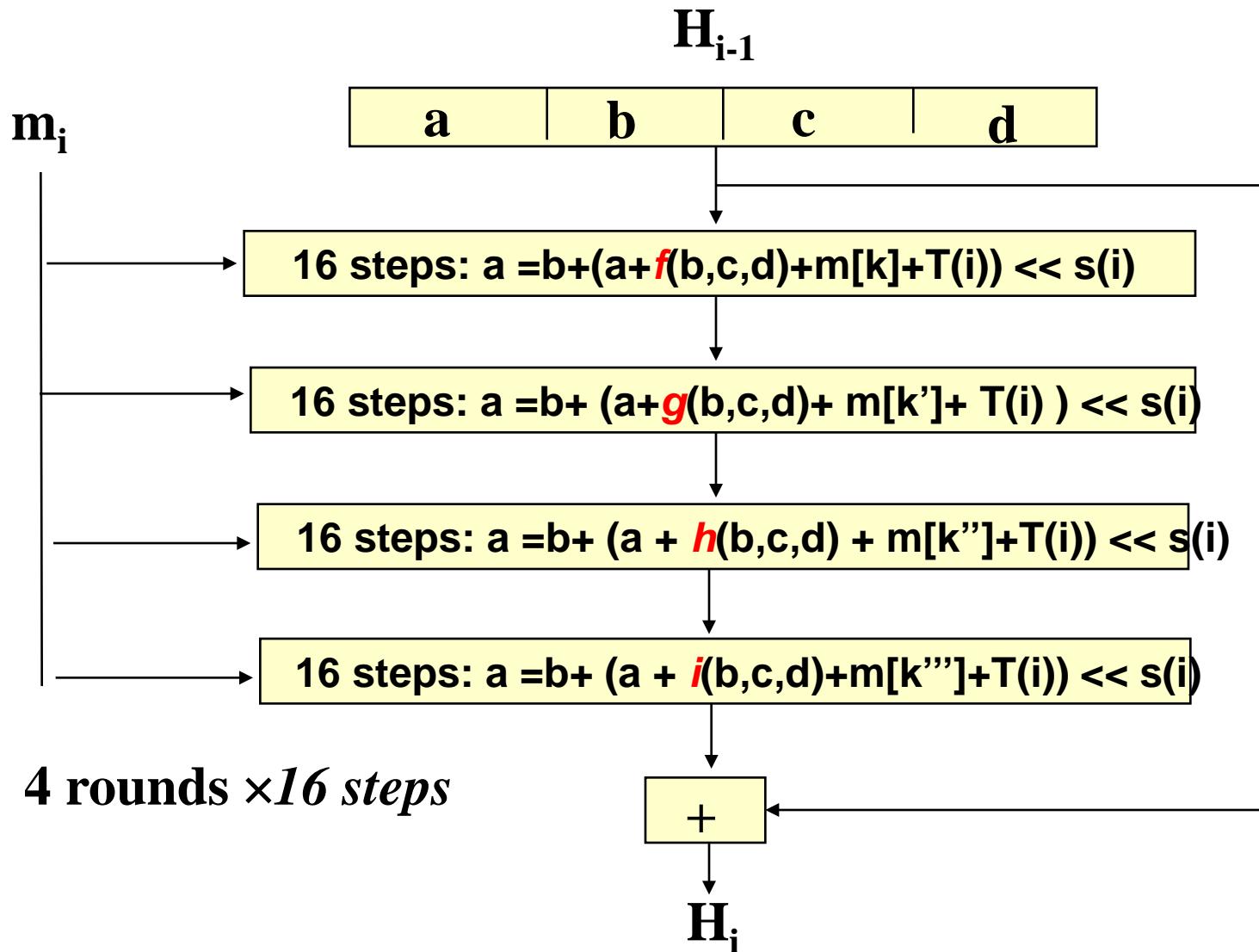


# MD5 Overview



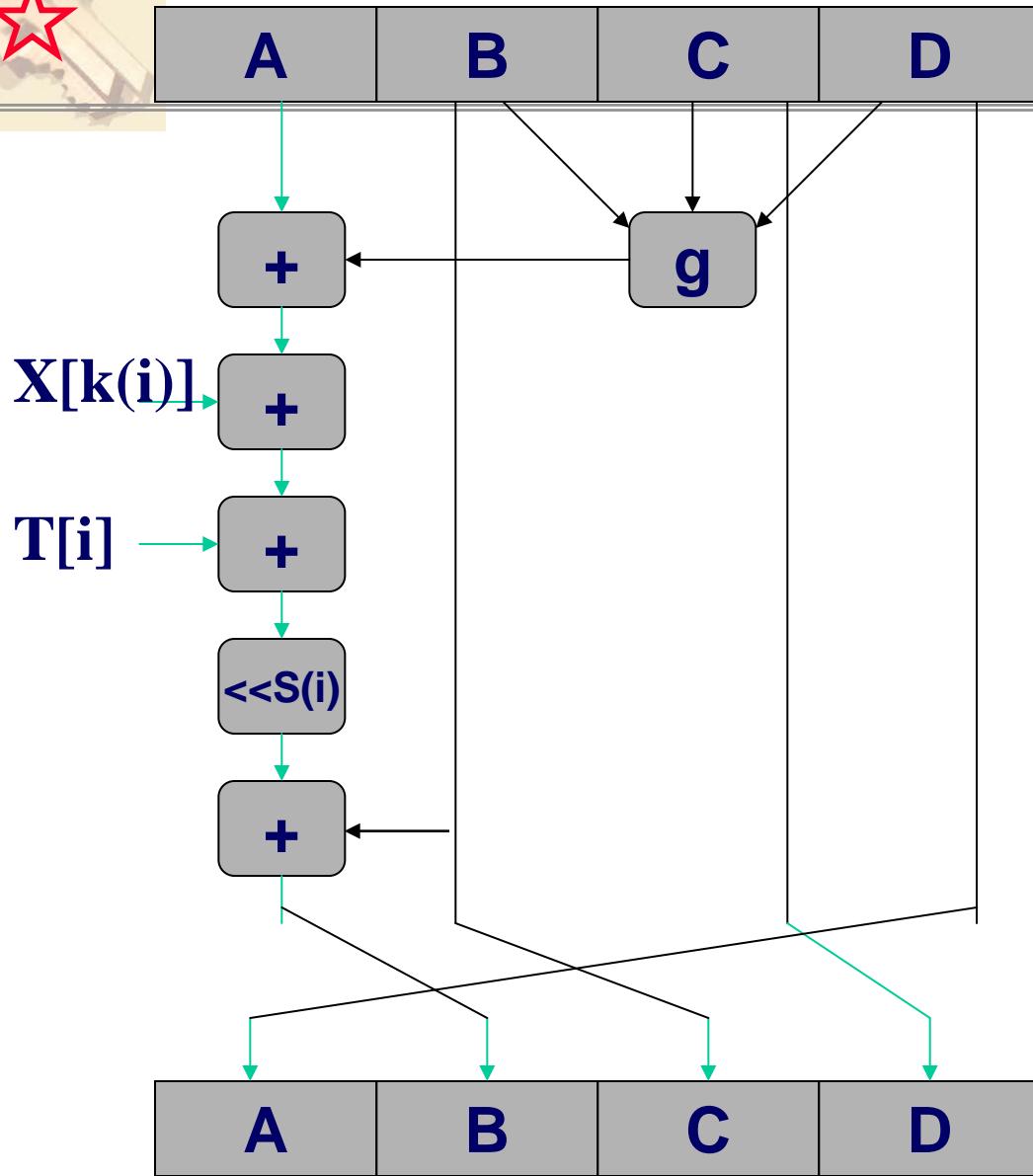


# md5 compress function





# md5 step



Function  $g(b,c,d)$

1.  $f(X,Y,Z) = XY \vee \text{not}(X) Z$
2.  $g(X,Y,Z) = XZ \vee Y \text{ not}(Z)$
3.  $h(X,Y,Z) = X \text{ xor } Y \text{ xor } Z$
4.  $i(X,Y,Z) = Y \text{ xor } (X \vee \text{not}(Z))$

$$k_1(i) = i$$

$$k_2(i) = (1+5i) \bmod 16$$

$$k_3(i) = (5+3i) \bmod 16$$

$$k_4(i) = 7i \bmod 16$$



# SHA-1



- 160-bit hash code, five 32-bit variables
- 4 rounds, every round has 20 steps
- 4 functions:  $f, h, g, h$ , the same as in MD4
- Message expansion: each 16-word (512-bit) message block is expanded to an 80-word block

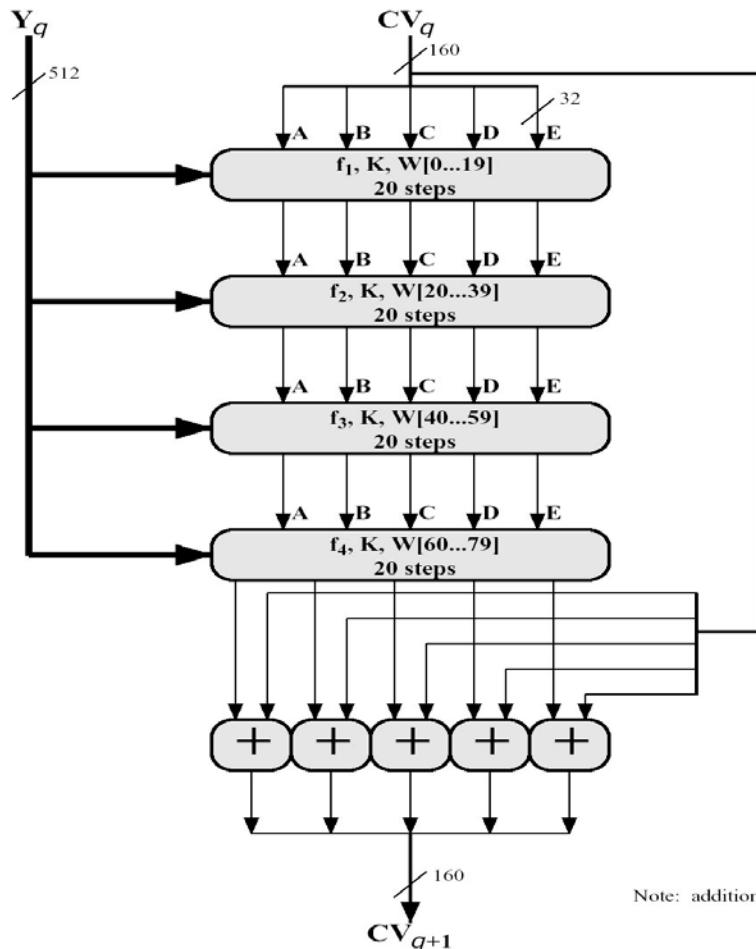
$W(t)=M(t)$   $t=0, \dots, 15$ ; for  $t=16, \dots, 79$ :

$W(t)=\text{rot}^1( w(t-16) \oplus w(t-3) \oplus w(t-8) \oplus w(t-14) )$

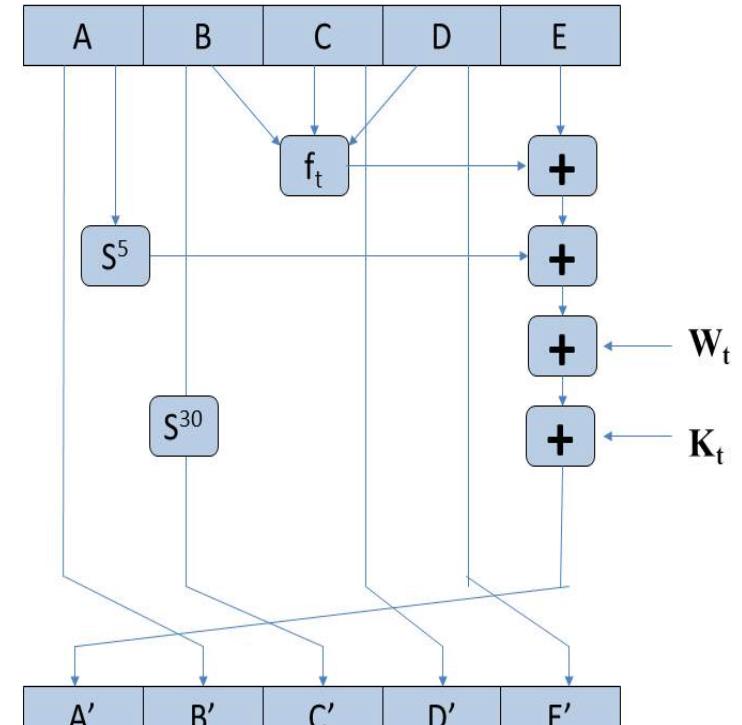
- modification in rotation ( $\text{rot}^1$ : from SHA-0)
- Same padding as MD4
- RFC3174, FIPS 180-1 (1995)



# SHA-1 compress function



4 rounds  
× 20 steps



SHA-1 step



# SHA-1 step functions

- 4 rounds  $\times$  20 steps:  $0 \leq t < 80$

$$E \leftarrow E + f_i(t, B, C, D) + (A \ll 5) + W[t] + K[t]$$

$$B \leftarrow B \ll 30$$

$$(A, B, C, D, E) \leftarrow (A, B, C, D, E) \gg 32$$

$$- f(t, B, C, D) = (BC) \oplus (\neg BD) \quad (\text{ch}) \quad 0 \leq t < 20$$

$$K[t] = 2^{30} \times \sqrt{2}$$

$$- h(t, B, C, D) = B \oplus C \oplus D \quad (\text{parity}) \quad 20 \leq t < 40$$

$$K[t] = 2^{30} \times \sqrt{3}$$

$$- g(t, B, C, D) = (BC) \oplus (BD) \oplus (CD) \quad (\text{maj}) \quad 30 \leq t < 60$$

$$K[t] = 2^{30} \times \sqrt{5}$$

$$- h(t, B, C, D) = B \oplus C \oplus D \quad 60 \leq t < 80$$

$$K[t] = 2^{30} \times \sqrt{10}$$



# SHA-224, 256, 384, 512



SHA	length	Message length	unit	IV	Message block	constants	Steps
-1	160 =5x32	<2 <sup>64</sup>	32-bit	0123...	512	5a827999 6ed9eba1 8f1bbcdcc ca62c1d6	4X20
-256	256 =8x32	<2 <sup>64</sup>	32-bit	Sqrt(p <sub>i</sub> ) i=1-8	512	P <sub>i</sub> <sup>1/3</sup> i=1-64	64
-224	224 Truncate SHA-256	<2 <sup>64</sup>	32-bit	Sqrt(p <sub>i</sub> ) i=1-8	512	P <sub>i</sub> <sup>1/3</sup> i=1-64	64
-384	384 Truncate SHA-512	<2 <sup>128</sup>	64-bit	Sqrt(p <sub>i</sub> ) i=1-8	1024	P <sub>i</sub> <sup>1/3</sup> i=1-80	80
-512	512 =8x64	<2 <sup>128</sup>	64-bit	Sqrt(p <sub>i</sub> ) i=1-8	1024	P <sub>i</sub> <sup>1/3</sup> i=1-80	80

FIPS 180-2 [NIST 2002]



# SHA-256 functions

Non-linear functions used:

$$Maj(B,C,D) = (BC) \oplus (BD) \oplus (CD) \quad (maj)$$

$$ch(B,C,D) = (BC) \oplus (\neg BD)$$

$$\Sigma_0(x) = rotr^2(x) + rotr^{13}(x) + rotr^{22}(x)$$

$$\Sigma_1(x) = rotr^6(x) + rotr^{11}(x) + rotr^{25}(x)$$

$$\sigma_0(x) = rotr^7(x) + rotr^{18}(x) + shr^3(x)$$

$$\sigma_1(x) = rotr^{17}(x) + rotr^{19}(x) + shr^{10}(x)$$

Message expansion

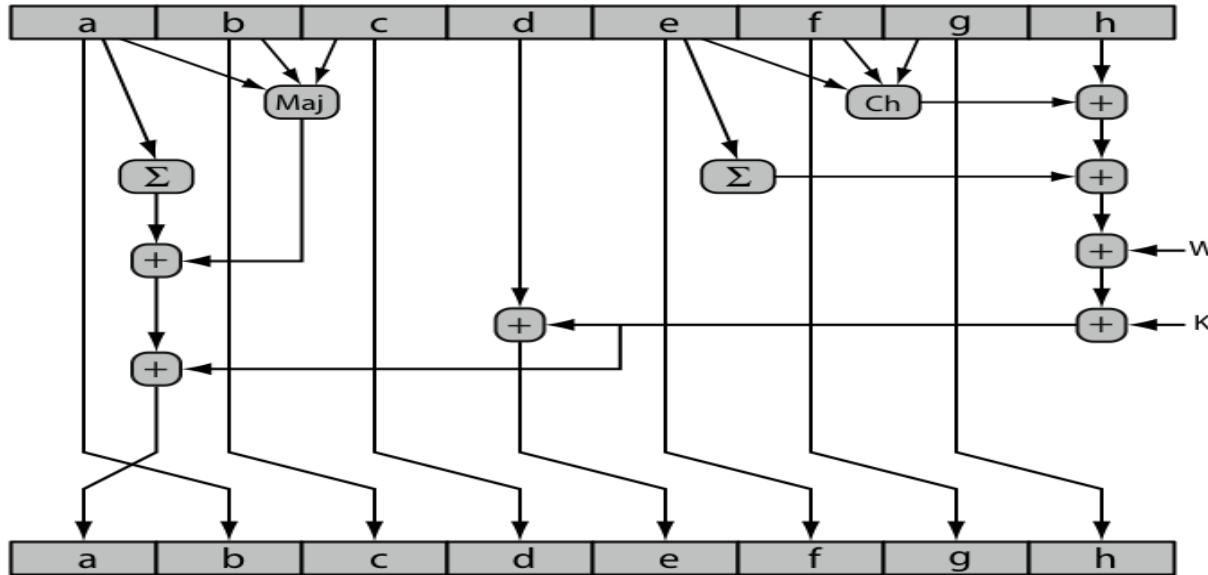
$$W(t) = M(t) \quad t=0, \dots, 15$$

$$W(t) = \sigma_1 W(t-2) + W(t-7) + \sigma_0 W(t-15) + W(t-16) \quad t=16, \dots, 63$$

Operations on 32-bit words



# SHA-256



8-block  
Feistel  
structure  
 $+ \text{ mod } 2^{32}$

- $(a, b, c, d, e, f, g, h) = (h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7)$
- **64 steps:**  $0 \leq t < 64$

$$T1 = h + \Sigma_1(e) + Ch(e, f, g) + K(t) + W(t)$$

$$T2 = \Sigma_0(a) + maj(a, b, c)$$

$$h = g; \quad g = f; \quad f = e; \quad e = d + T1$$

$$d = c; \quad c = b; \quad b = a; \quad a = T1 + T2$$

- $(h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7) = (a, b, c, d, e, f, g, h) + (h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7)$  DM



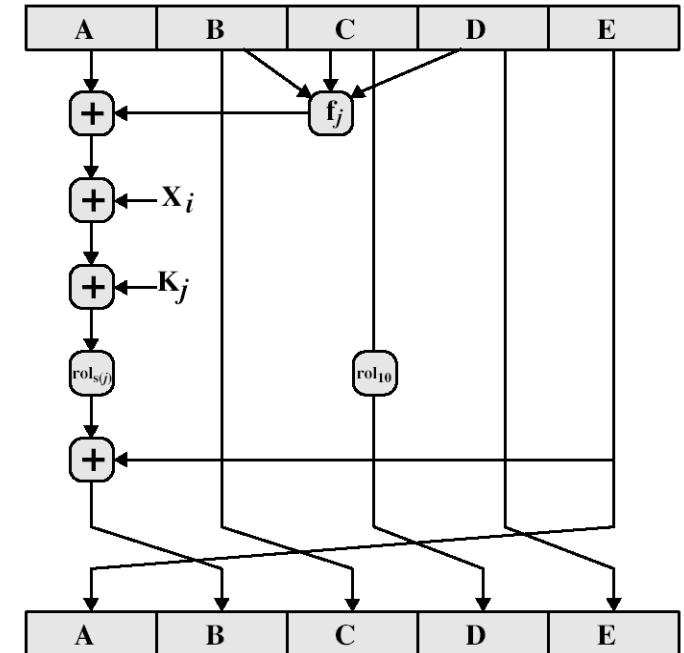
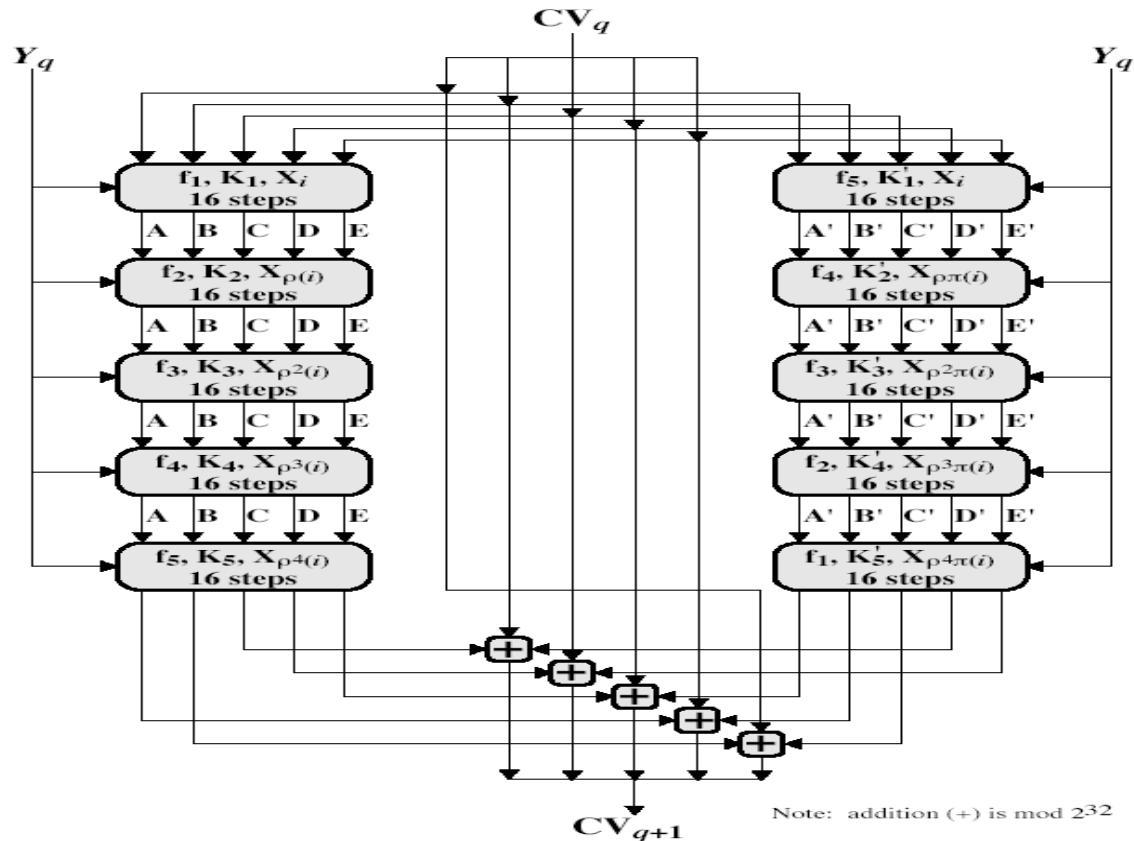
# RIPEMD



- RipeMD-160 [Bosselaers-Dobbertin Preneel,97]
  - compression function maps 21-word input (5-word chaining variable, 16 words of 32-bit message block) to 5-word output
$$(a,b,c,d,e)_{i-1};(m_0,m_1,\dots m_{15})_i \rightarrow (a,b,c,d,e)_i$$
  - “Parallel 5-block MD5”
  - 160-bit hash code, comparable with SHA-1
- RipeMD-128: “Parallel MD5”
- RipeMD: “Parallel MD4”



# RIPEMD-160



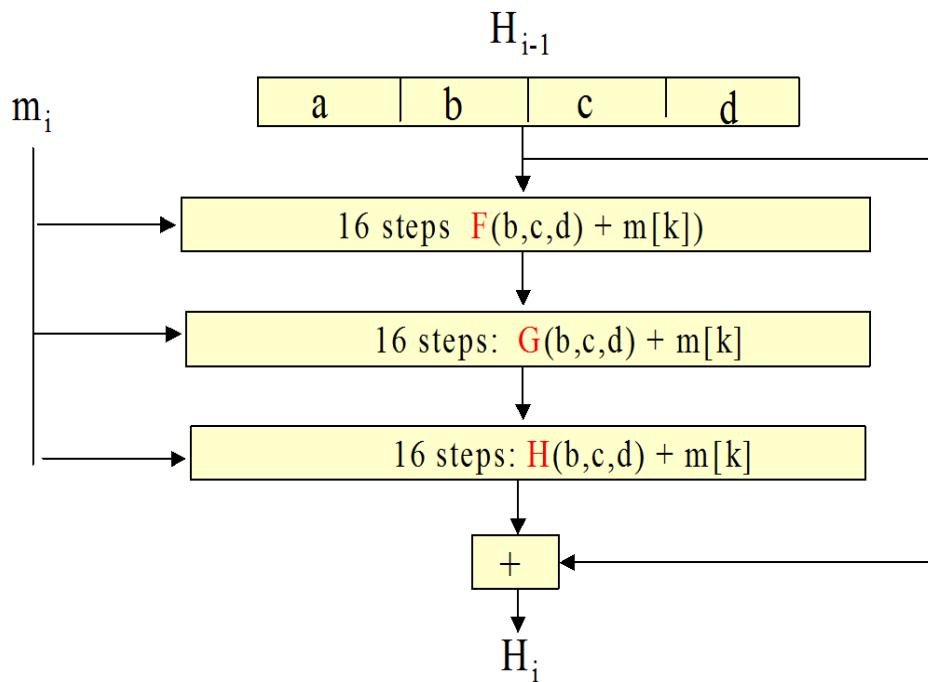
**Parallel: 5 rounds X16 steps // 5 rounds X16 steps**

160-bit IV=(A,B,C,D,E)=(67452301,efcdab89,98badcfe,10325476,c3d2e1f0)

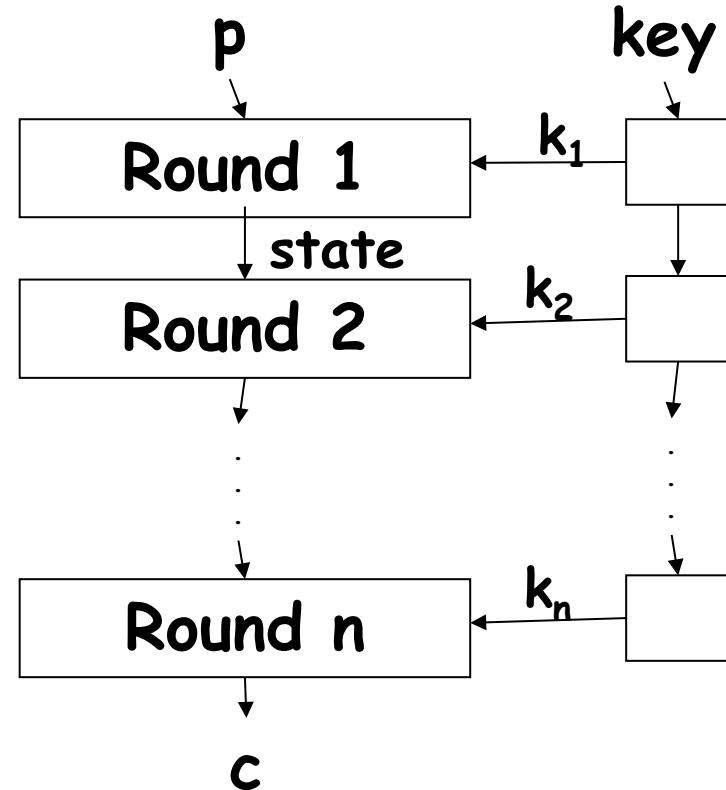


# Attack on Hash vs attack on cipher

md4



cipher



Hash: find different  $m_i, m'_i$  so  $\Delta H=0$

Cipher: choose  $(p, c)$  to find subkey  $k_i$

Message expansion --- key schedule



# MD4



- Dobbertin: Collision ( $2^{22}$ ) [96], collision for meaningful messages [96], reverse for first 32 steps (of total 48) [98]
- Wang et.al [04-05]:
  - Collision on compress function : Complexity  $2^2 \sim 2^6$
  - Target (2nd pre-image) attack with success probability  $2^{-56}$
- Pre-image: [Zhong-Lai,2011] Complexity  $2^{95}$



# MD5



- Rivest [92]: as an improvement of MD4, most widely used.
- Boer & Bosselaers [93]: **free-start collision** (pseudo collision: same message, different IV ) on **compress** function

$$md5(H_0, M) = md5(\textcolor{red}{H}_1, M)$$

- Dobbertin [96]: **semi free-start collisions** (different messages, chosen IV) on **compress** function: Find  $H, M, \textcolor{red}{M}' \neq M$ , but

$$md5(H, M) = md5(H, \textcolor{red}{M}')$$

- Wang et.al [Crypto 04, Eurocrypt 05]: **collision attack with complexity  $2^{37}(2^{39}, 2^{32})$**

$$MD5(H_0, M) = MD5(H_0, \textcolor{red}{M}')$$



# Broken hash ?



What is “**broken**”?

- **Academically** broken: attacks with complexity less than brute-force;
- **Practically** broken: user or vendor have concern to use it to protect their data;
- **Psychologically** broken: just a collision pair.
- Example:
  - block cipher DES: [Biham 91], [NIST 97]
    - (7 years from academically broken to practically broken)
  - Hash MD5: [Boer 93], [Wang.. 04]
    - (12 years from academically broken to practically broken)
  - single-length DES hash: 64-bit
    - (practically broken but never academically broken)
- a collision pair can lead to lots of things by clever people



# Broken examples

- Block cipher DES: [Biham 91], [NIST 97]
  - Diff. att (91):  $2^{47}$  - **academically broken**
  - Search engine (95): hours, weeks - **practically broken**
- Hash MD5: [Boer 93], [Wang 04]
  - FS-collision (93) - **academically broken**
  - Collision (04) – **Psychologically (practically?) broken**
- Single-length DES hash:
  - 64-bit: **practically broken** from beginning
  - Never been **academically broken**



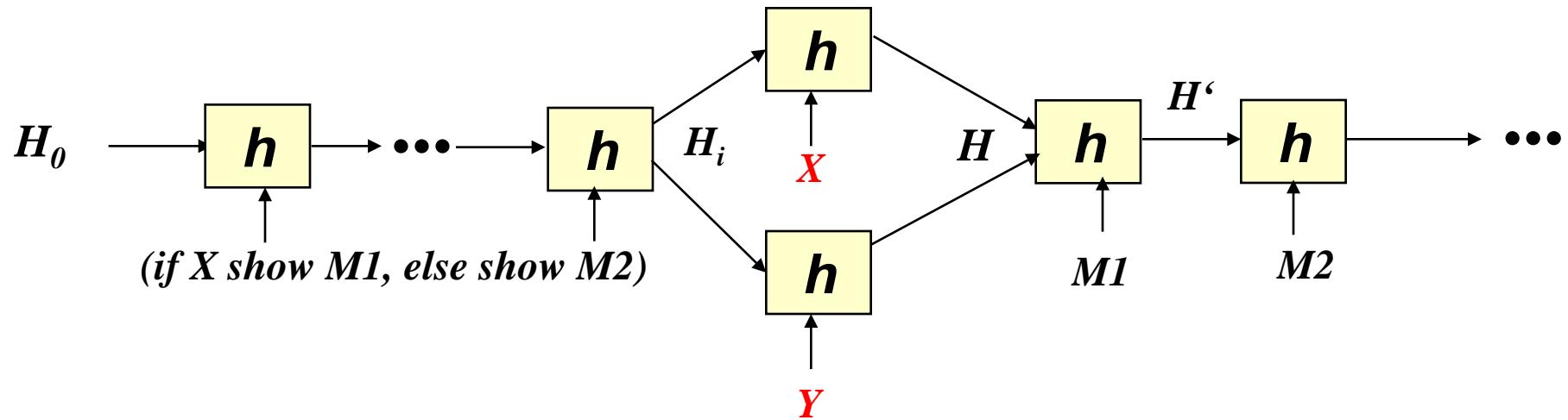
# Meaningful Collisions for MD5



- Stefan Lucks and Magnus Daum (Eurocrypt'05 Rump Session)
- <http://th.informatik.unimannheim.de/people/lucks/HashCollisions/>
- <http://www.cits.rub.de/MD5Collisions/>
- 2 postscript files:  
M1: a recommendation letter for Alice  
M2: an order letter for Alice's privilege
- Both letters have the same signature because of  $\text{MD5}(M1)=\text{MD5}(M2)$
- The Boss will sign M1 (harmless)
- Alice can then use M2



# Document collision with MD5



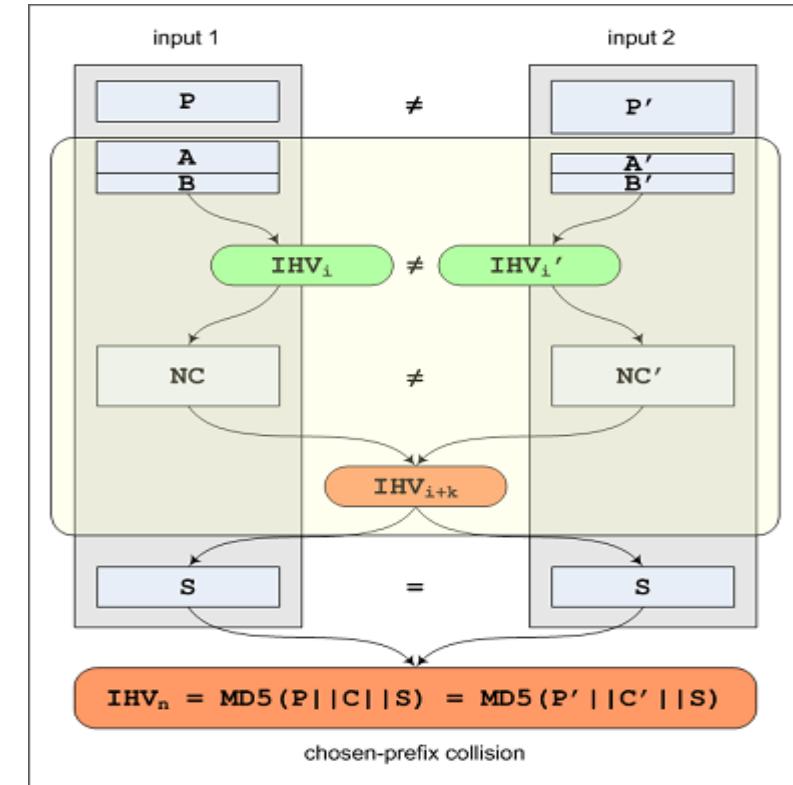
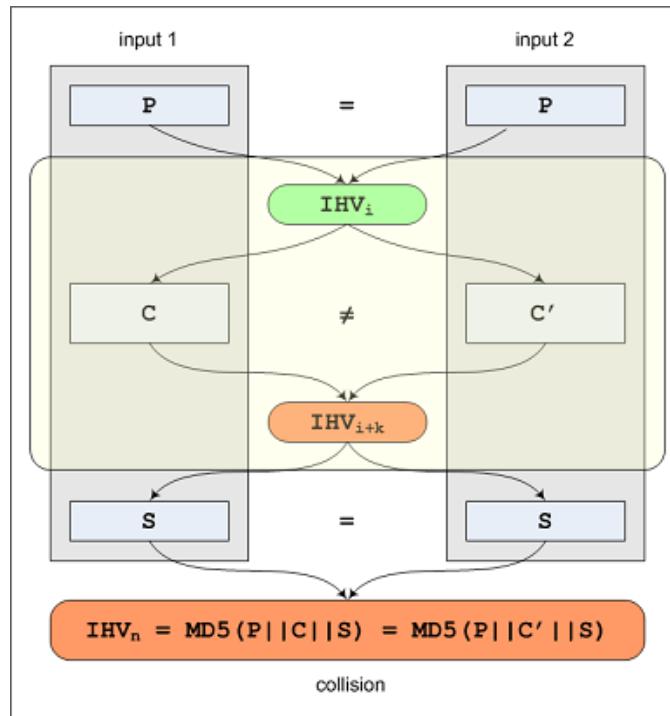
- Fixed  $H_0$ , select prefix message, from the resulting  $H_i$ , find colliding messages  $X, Y$ ; then attach  $M1$  and  $M2$ .
- (instruction,  $X, M1, M2$ )
- (instruction,  $Y, M1, M2$ )
- Have same hash code (signature)



# MD5 collision – chosen-prefix collision



- “rogue certificates” [M. Stevens,,09] <http://eprint.iacr.org/2009/111>
  - 2 certificates with different data fields (especially CA=TRUE/FALSE) and public-keys, but with same MD5 hash code.
  - Chosen free-start collision: comp.= $2^{16}$





## real certificate

## rogue CA certificate



		header				header			
		version number "3" serial number "643015"				serial-number "65" version number "3"			
		signature algorithm "MD5 with RSA"				signature algorithm "MD5 with RSA"			
issuer		country "US"		country "US"		country "US"		country "US"	
		organization "Equifax Secure Inc."							
		common name "Equifax Secure Global eBusiness CA-1"		common name "Equifax Secure Global eBusiness CA-1"		common name "Equifax Secure Global eBusiness CA-1"		common name "Equifax Secure Global eBusiness CA-1"	
		validity "from 3 Nov. 2008 7:52:02 to 4 Nov. 2009 7:52:02"		validity "from 31 Jul. 2004 0:00:00 to 2 Sep. 2004 0:00:00"		validity "from 31 Jul. 2004 0:00:00 to 2 Sep. 2004 0:00:00"			
		subject		subject		subject		subject	
		country "US"		country "US"		country "US"		country "US"	
		organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"		organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"		organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"		organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"	
		organizational unit "GFI11029001"		organizational unit "See www.rapidssl.com/resources/cps (c) 08"		organizational unit "Domain Control Validated - RapidSSL(R)"		organizational unit "Domain Control Validated - RapidSSL(R)"	
		common name "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"							
		public key algorithm "RSA"							
		modulus (2048 bits)							
		header		header		header		header	
		B2D3 2581AA28E878B1E5		5AD5C0F36576E29		5AC5450B36BB01D1		53AAC3088F6FF84F	
		5F06410E6B4C807		17000000 5BF0D6B1C7B9C829		53AAC3088F6FF84F		3EB7874411DC6080	
		3EB7874411DC6080		DF9255F8731854		3562CD89AFICA686		93C59FD046C46086	
		93C59FD046C46086		3562CD89AFICA686		1AC95B3C9637C0ED		1AC95B3C9637C0ED	
		67EFBBEC08B9C50		67EFBBEC08B9C50		67EFBBEC08B9C50		67EFBBEC08B9C50	
		birthday bits (96)							
		block 9		1 <sup>st</sup> near collision block		block 10		2 <sup>nd</sup> near collision block	
		block 11		3 <sup>rd</sup> near collision block		block 12		block 13	
		(identical)		(identical)		(identical)		(identical)	
		key usage "..."							
		subject key identifier "..."							
		crl distribution points "..."							
		authority key identifier "..."							
		extended key usage "..."							
		basic constraints "CA = FALSE"		basic constraints "CA = FALSE"		basic constraints "CA = TRUE"			
		signature algorithm "MD5 with RSA"							
		signature		signature		signature		signature	
		A721028BD10E8A280 7725FD4360158FEC		EF9047D4E4421526 111CCDC23C1029A9		A721028BD10E8A280 7725FD4360158FEC		EF9047D4E4421526 111CCDC23C1029A9	
		B6DFA8577591DAE5 2BB390451C306356		3F8AD950FAD586C C065AC6657DE1CC6		B6DFA8577591DAE5 2BB390451C306356		3F8AD950FAD586C C065AC6657DE1CC6	
		763BEF5000E8E45CE 7F4C90EC2BC6CDB3		B48F62D0FEB7C526 7244EDF6985BABC8B		763BEF5000E8E45CE 7F4C90EC2BC6CDB3		B48F62D0FEB7C526 7244EDF6985BABC8B	
		D195FDA08B8E6846 B175C8EC1DBF187A		94F1AA5378A245AE 54EAD19E74C87667		D195FDA08B8E6846 B175C8EC1DBF187A		94F1AA5378A245AE 54EAD19E74C87667	
		(identical)		(identical)		(identical)		(identical)	
		CA=False		CA=True					

From <http://www.win.tue.nl/hashclash/rogue-ca/>



# More results

- **RipeMD.** [Wang-Lai-Feng-Cen-Yu, crypto04, eurocrypt05]
  - Collision pair of compress function, complexity  $2^{19}$
  - target attack of probability  $2^{-125}$
- **SHA-0.** [Wang-Yu-Yin, Crytpo05]
  - Collision pair of SHA-0: Complexity  $2^{39}$
  - Semi free-start collision of compress function.
  - target attack of probability  $2^{-107}$ .
- **SHA-1.** [Wang-Yin-Yu, Crytpo05]
  - Collision pair of first 58 steps, Complexity  $2^{33}$
  - Collision attack on SHA-1 with complexity  $2^{69}$  ( $2^{66}, 2^{63}$ )
  - Collision using new path,  $2^{52}$  [AC09]



# SHA-3

- 2007 NIST decided to develop one or more additional hash functions through a public competition.
- Call for a New Cryptographic Hash Algorithm (SHA-3) Family on November 2, 2007
- 2009. First Hash Function Candidate Conference
- 2010. Second Hash Function Candidate Conference, August 23-24, 2010
  - finalist candidates
- 2012. Final Hash Function Candidate Conference
  - final selection , draft standard
- 2012.10 NIST selected Keccak as SHA-3
- **2014.4.7 NIST Draft FIPS 202, SHA-3 Standard**



# Second Round Candidates



- **BLAKE** -- Jean-Philippe Aumasson
- Blue Midnight Wish -- Svein Johan Knapskog
- CubeHash -- D. J. Bernstein
- ECHO -- Henri Gilbert
- Fugue -- Charanjit S. Jutla
- **Grøstl** -- Lars Ramkilde Knudsen
- Hamsi -- Ozgul Kucuk
- **JH** -- Hongjun Wu
- **Keccak** -- Joan Daemen
- Luffa -- Dai Watanabe
- Shabal -- Jean-Francois Misarsky
- SHAvite-3 -- Orr Dunkelman
- SIMD -- Gaetan Leurent
- **Skein** -- Bruce Schneier



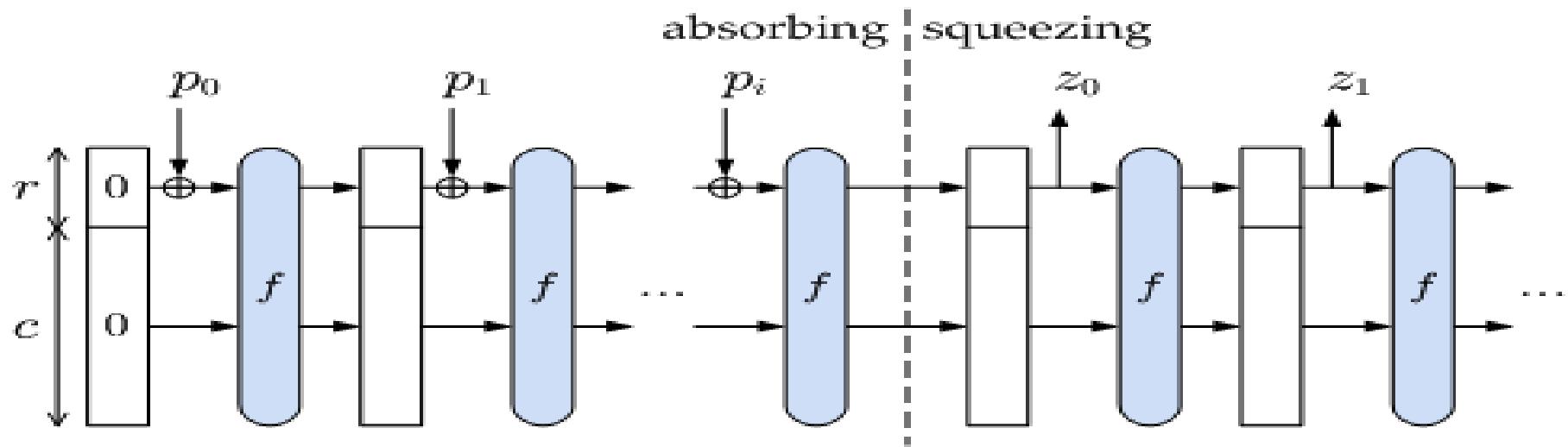
# SHA-3 candidates

	Block cipher	Permutation	MD/HAIFA
Blake			HAIFA
BMW	PGV variant		MD
Cubehash		Sponge	
ECHO			HAIFA
Fugue		Sponge	
Grøstl		2-permutation	MD
Hamsi			
JH			JH-specific
Keccak		Sponge	
Luffa		Sponge	
Shabal		Sponge	
Shavite-3	Davies-Meyer		HAIFA
SIMD	PGV variant		MD
Skein	Davies-Meyer		MD/Tree

From Bart Preneel talk, 2010.10



# Sponge Construction



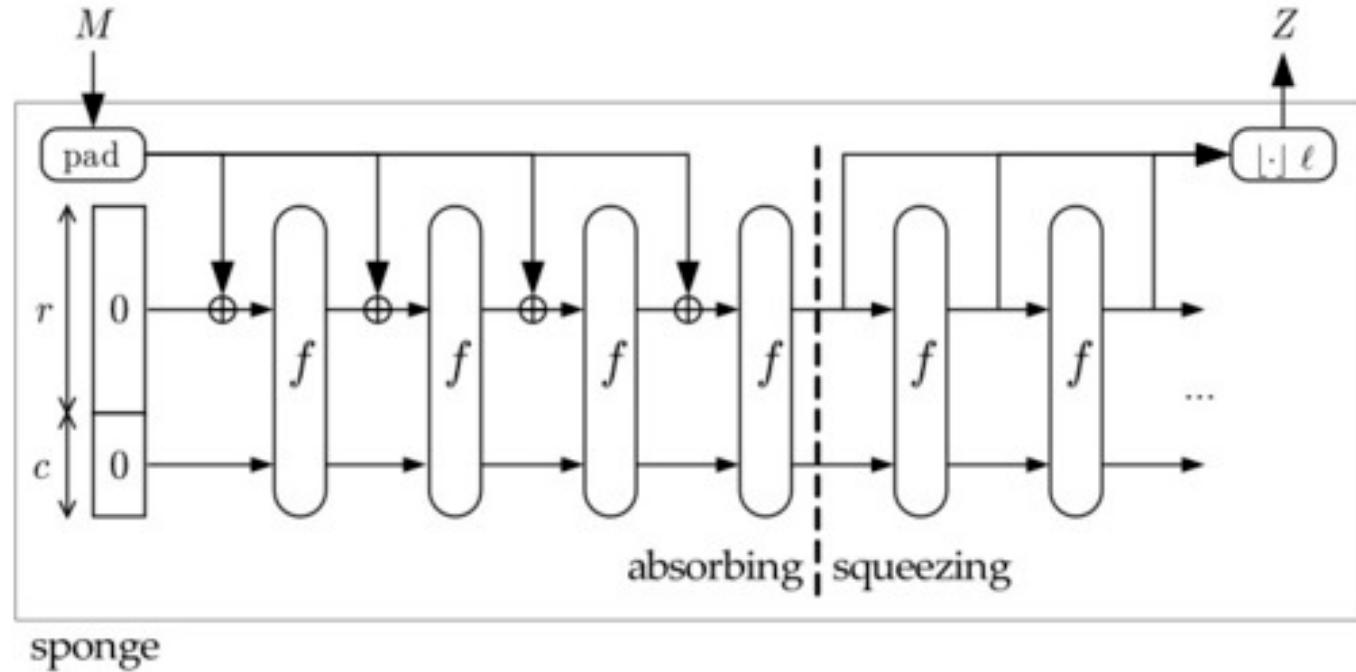
The sponge construction (larger)

- $(p_0, \dots, p_i)$  input (message)
- $(z_0, z_1, \dots)$  output (hash code)
- $f$  can be any transformation (permutation)
- SHA-3 (Keccak has this form)



# SHA-3

f is a permutation  
of 1600 bits  
Capacity  
 $c=2X\text{Hash}$



- Hash-224 bits:  $r = 1152$  and  $c = 448$
- Hash-256 bits:  $r = 1088$  and  $c = 512$
- Hash-384 bits:  $r = 832$  and  $c = 768$
- Hash-512 bits:  $r = 576$  and  $c = 1024$



# Sponge Construction



- Sponge Construction – based on **random permutation** is **different from**
- Merkle-Damgard construction – based on **one-way compress function**.
- a random sponge can **only be distinguished** from a random oracle due to inner collisions [Bertoni07]
- the sponge construction is **indifferentiable** from a random oracle when being used with a random transformation/permuation. [Bertoni08]



# SHA-3: Keccak



KECCAK- $f[b]$  is an iterated permutation, consisting of a sequence of  $n_r$  rounds R, indexed with  $i_r$  from 0 to  $n_r - 1$ . A round consists of five steps:

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta, \text{ with}$$

$$\theta : a[x][y][z] \leftarrow a[x][y][z] + \sum_{y'=0}^4 a[x-1][y'][z] + \sum_{y'=0}^4 a[x+1][y'][z-1],$$

$$\rho : a[x][y][z] \leftarrow a[x][y][z - (t+1)(t+2)/2],$$

with  $t$  satisfying  $0 \leq t < 24$  and  $\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$  in  $\text{GF}(5)^{2 \times 2}$ ,  
or  $t = -1$  if  $x = y = 0$ ,

$$\pi : a[x][y] \leftarrow a[x'][y'], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix},$$

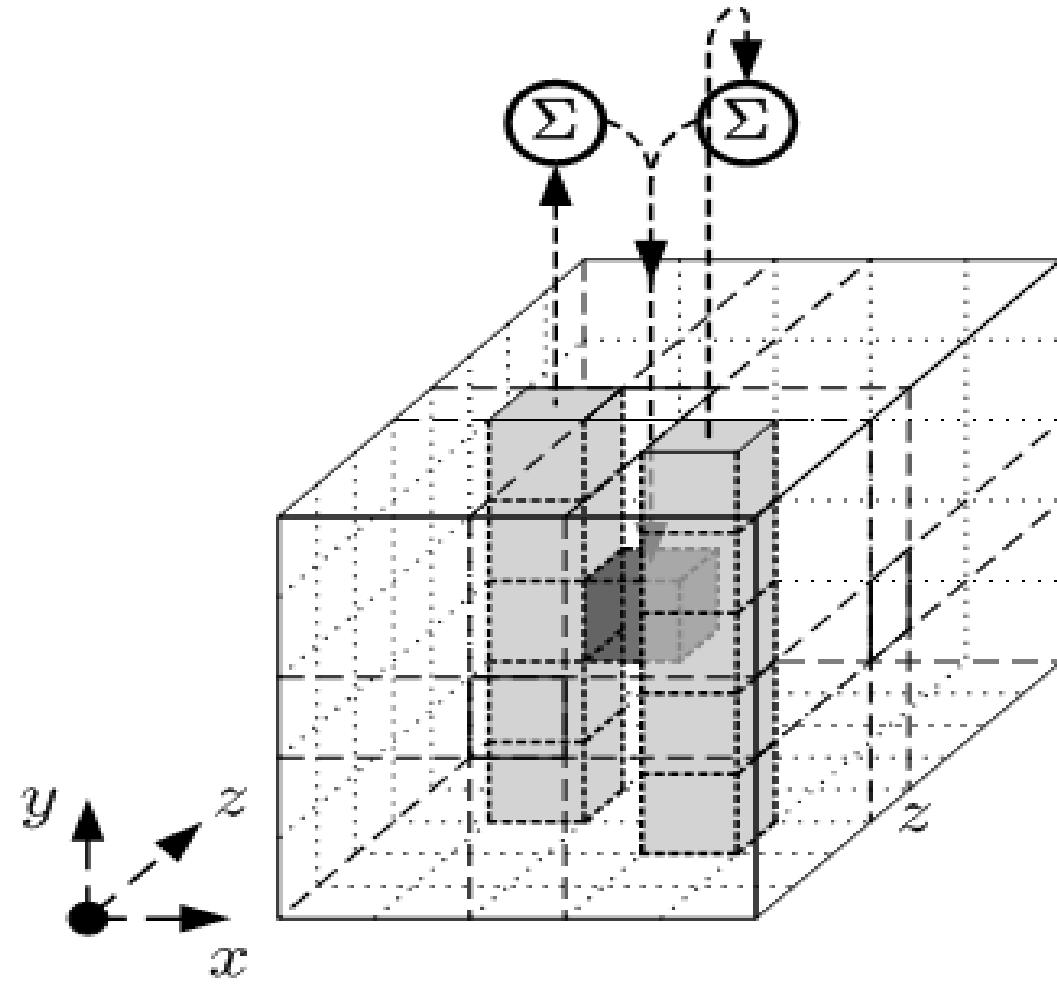
$$\chi : a[x] \leftarrow a[x] + (a[x+1] + 1)a[x+2],$$

$$\iota : a \leftarrow a + \text{RC}[i_r].$$

SHA-3:  $b=1600$  bits = 64 slices of  $5 \times 5$  bits



# Function $\theta$





# DoP: One-way functions

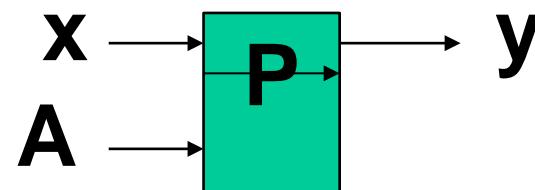


Besides D-log

1. cipher  $E(P,K)=C$ ,  $a=\text{constant}$ ;  $f(X)=E(a,X)$  is one-way if  $E$  is ideal.

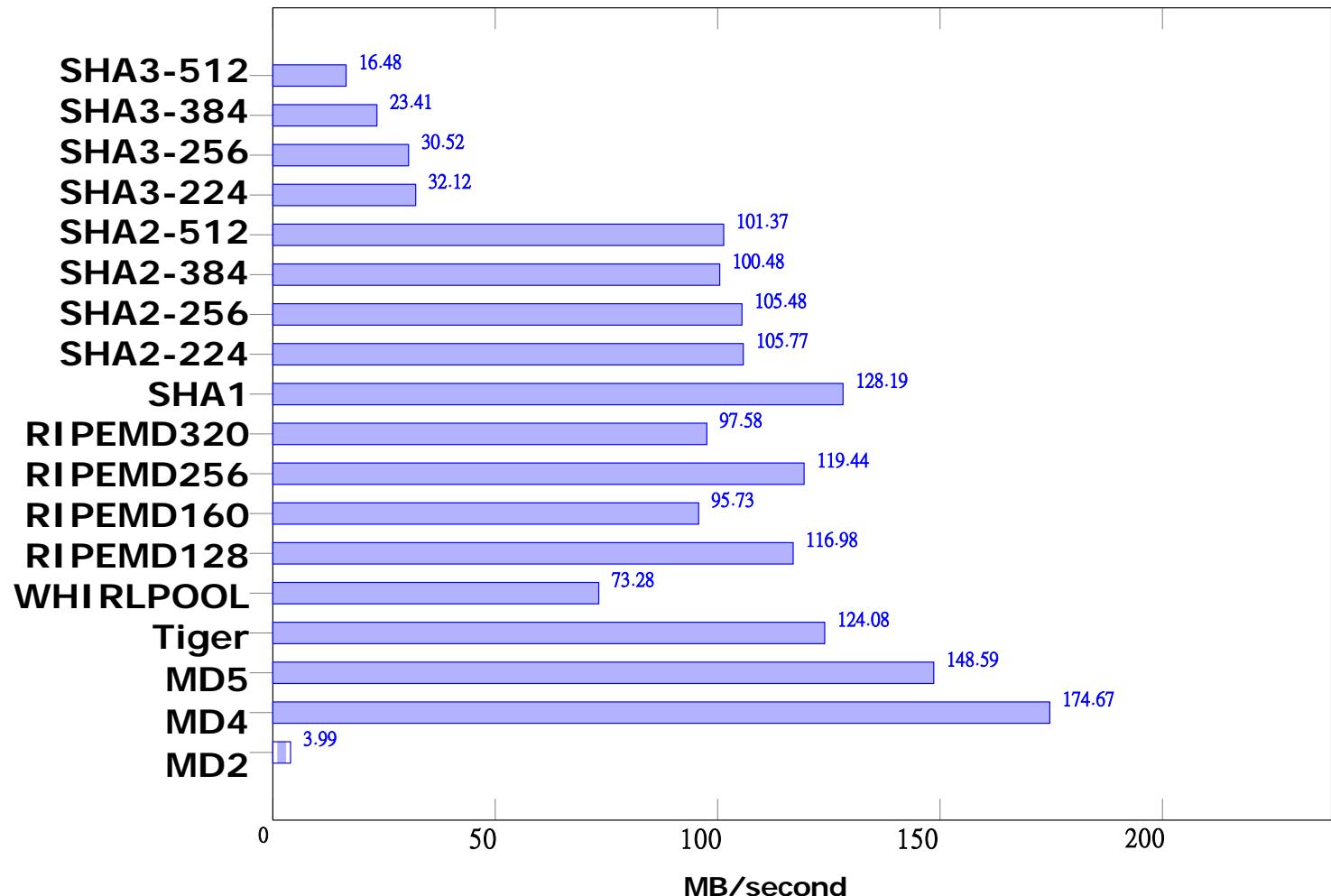
2. Permutation  $P$ ,  $f(X)=P(X)+X$  is one-way

3. Permutation  $P(x,X)=(y,Y)$ ,  $y=f(x)=P^t(x,A)$  is one-way





# HASH Performance





# One-way functions

- **Oneway function**  $f: X \rightarrow Y$ , given  $x$ , easy to compute  $f(x)$ ; but for given  $y$  in  $f(X)$ , it is hard to find  $x$ , s.t.,  $f(x)=y$ .
  - $\text{Prob}[ f(A(f(x))=f(x)) ] < 1/p(n)$  (TM definition, existence unknown)
  - Example: hash function, discrete logarithm;
- **Keyed function**  $f(X,Z)=Y$ , for known key  $z$ , it is easy to compute  $f(.,z)$ 
  - Block cipher
- **Keyed oneway function**:  $f(X,Z)=Y$ , for known key  $z$ , it is easy to compute  $f(.,z)$  but for given  $y$ , it is hard to  $x,z$ , s.t.,  $f(x,z)=y$ .
  - MAC function: keyed hash  $h(z,X)$ , block cipher CBC
- **Trapdoor oneway function**  $f_T(x)$ : easy to compute and hard to invert, but with additional knowledge  $T$ , it is easy to invert.
  - Public-key cipher; RSA:  $y=x^e \text{ mod } N$ ,  $T: N=p*q$



# MAC: Message authentication Code

- a MAC is a cryptographic checksum  
$$\text{MAC} = C_K(M)$$
  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- is a many-to-one function
  - potentially many messages have same MAC
  - but finding these needs to be difficult



# Requirements for MACs

- **Security** of MAC:

If the key  $k$  is unknown, it is difficult to find a new message with a valid MAC, even if many valid  $(M, C_k(M))$  are known.

The  $M$  in above  $(M, C_k(M))$  can be known or chosen.



# Construction of MAC



- based on CBC and CFB modes of a block cipher
  - MAA(Message Authenticator Algorithm)
    - ISO standard
    - relative fast in S/W
    - 32-bit result
- based on hash functions
  - Keyed Hash Functions
    - fast than other schemes
    - additional implementation effort is small
    - adopted in Kerberos and SNMP



# Keyed Hash Functions as MACs

- Create a MAC using a hash function rather than a block cipher
  - because hash functions are generally faster
  - not limited by export controls unlike block ciphers
- hash includes a key along with the message
- original proposal:  
**KeyedHash = Hash(Key | Message)**
  - some weaknesses were found with this
- Password recovery attack on APOP by MD5 collision.



# HMAC-NMAC

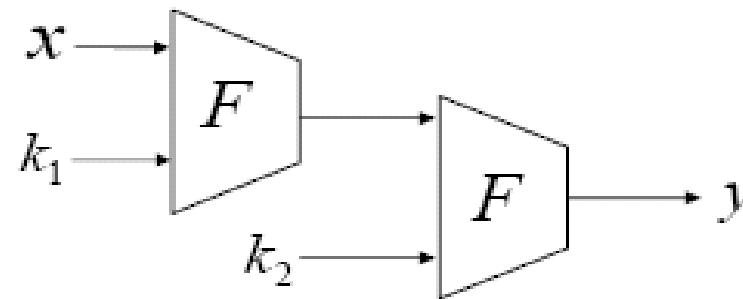
- HMAC (Internet standard RFC2104)
  - uses hash function on the message:  
$$\text{HMAC}_K = \text{Hash}[(K^+ \text{ XOR } \text{opad}) \parallel \text{Hash}[(K^+ \text{ XOR } \text{ipad}) \parallel M]]$$
  - where  $K^+$  is the key padded out to size
  - and opad, ipad are specified padding constants
  - any of MD5, SHA-1, RIPEMD-160 can be used
- NMAC= $H(k_2, H(k_1, x))$
- Essentially, Hash(Key | Message | Key)



# NMAC and HMAC

We assume the length of  $x$  is only one message block(after padding).

- $\text{NMAC}_{k_1, k_2}(x) = F_{k_2}(F_{k_1}^*(x))$



- $\text{HMAC}_k(x) = H_{iv}^*(k \oplus \text{OPAD} \parallel H_{iv}^*(k \oplus \text{IPAD} \parallel x)),$

