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

2016-04

Organization

- Week 1 to week 16 (2016-02-24 to 2016-06-08)
- 东上院502
- 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: ‘薛伟佳’ icelikejia@qq.com，'黄格仕' <huang.ge.shi@foxmail.com>
- Send homework to: laix@sjtu.edu.cn and to TAs

Rule: do not disturb others!

2

Contents

- Introduction -- What is security?
- Cryptography
  - Classical ciphers
  - Today's ciphers
  - Public-key cryptography
  - Hash functions/MAC
  - Authentication protocols
- Applications
  - Digital certificates
  - Secure email
  - Internet security, e-banking

Network security
- SSL
- IPSEC
- Firewall
- VPN

Computer security
- Access control
- Malware
- DDos
- Intrusion

Examples
- Bitcoin
- Hardware
- Wireless

3

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

4

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 Damgard. 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.

5

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**

6

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

7

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

8

☆ Hash-functions from block ciphers

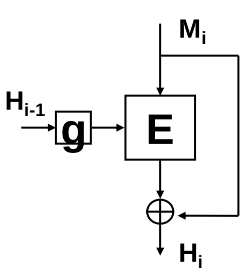
- ISO/IEC 10118-2
- Obtain a hash-function from an *m*-bit block cipher.
  - *Method 1* - hash-codes up to *m* bits long,
  - *Methods 2 & 3* - hash-codes up to 2*m* bits,
  - *Method 4* - hash-codes up to 3*m* bits long.

- Basic method: Davies-Meyer construction.
  - one-way function from a permutation:

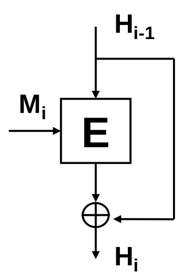
$$h(x,k) = e_k(x) \oplus x$$

9

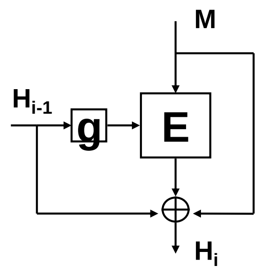
Single length



Matyas-Meyer-Oseas



Davies-Meyer



Miyaguchi-Preneel

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

10

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 × 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.

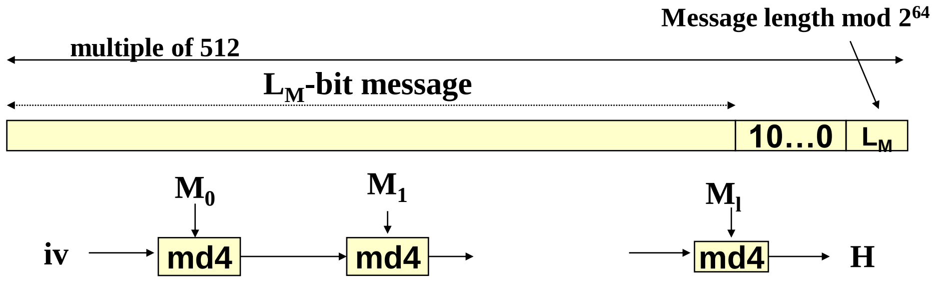
11

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

12

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.

13

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}

14

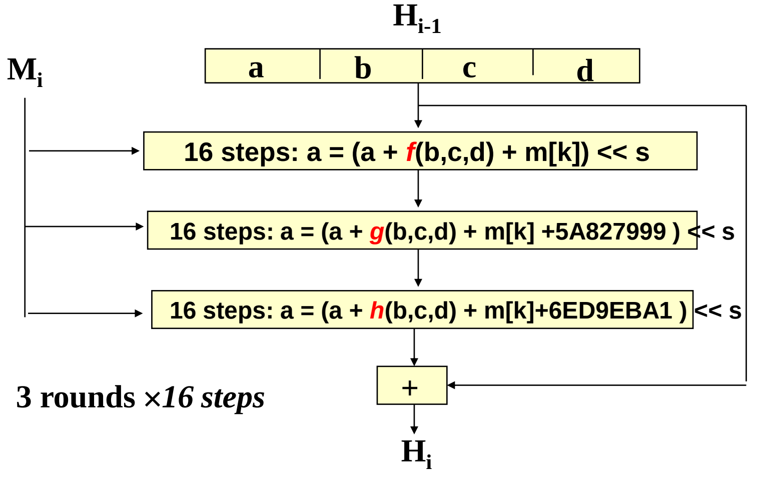
MD4 compress function

- 512-bit message block m_i=(m[0],m[1],…,m[15])
- then three 16-step rounds (A,B,C,D) ←(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,…)
  - A ←(A+*f*(B,C,D)+m[j])<<s(j),
  - (A,B,C,D) ←(D,A,B,C)
- Round 2: for j=0 to 15, // s(j)=(3,5,9,13,3,5,9,13,…) (step 16-31)
  - A ←(A+*g*(B,C,D)+m[j]+5A827999)<<s(j),
  - (A,B,C,D) ←(D,A,B,C)
- Round 3: for j=0 to 15, // s(j)=(3,9,11,15,3,9,11,15,…) step32-47)
  - A←(A+*h*(B,C,D)+m[j]+6ED9EBA1)<<s(j),
  - (A,B,C,D) ←(D,A,B,C)
- (H_1, H_2, H_3, H_4) ← (H_1+A, H_2+B, H_3+C, H_4+D)

- 5A827999 is $2^{1/2}$, 6ED9EBA1 is $3^{1/2}$

15

MD4 compress function

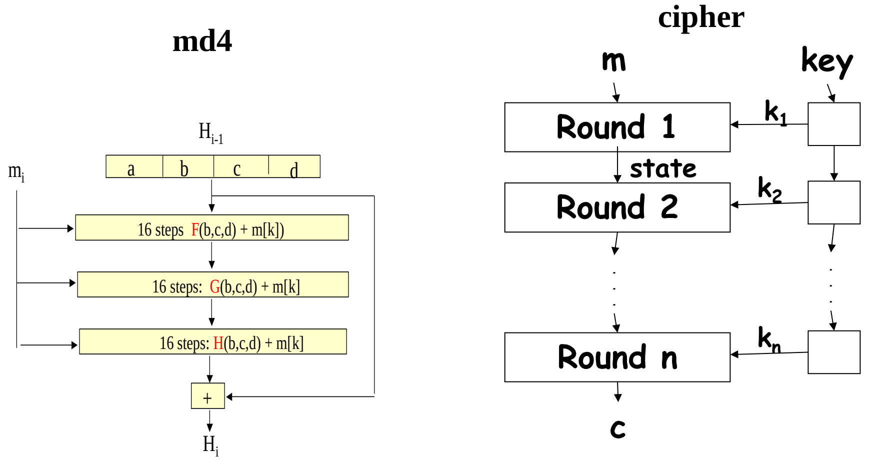


+ mod 2^{32}

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

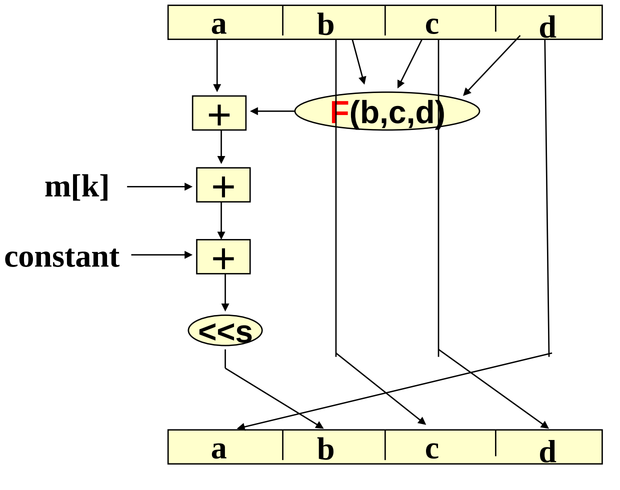
16

Hash-step vs cipher-round



17

MD4 step function



4-block
feistel structure

+ mod 2^{32}

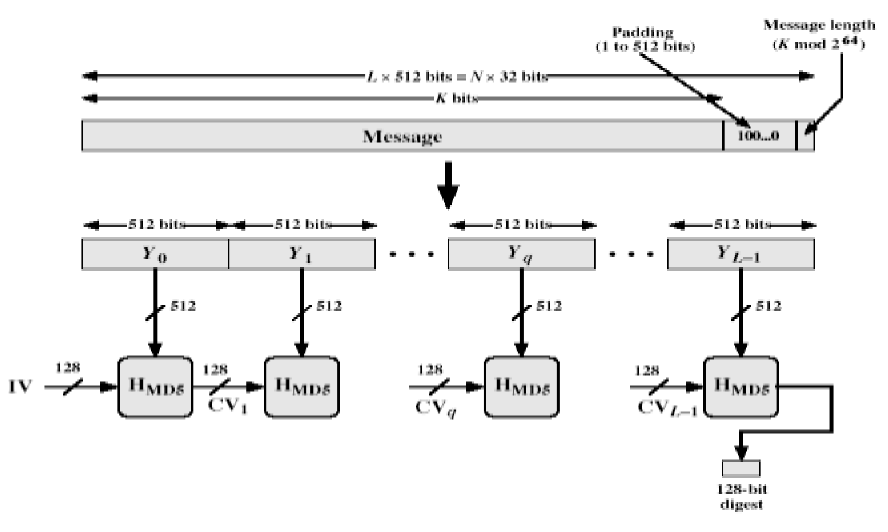
18

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

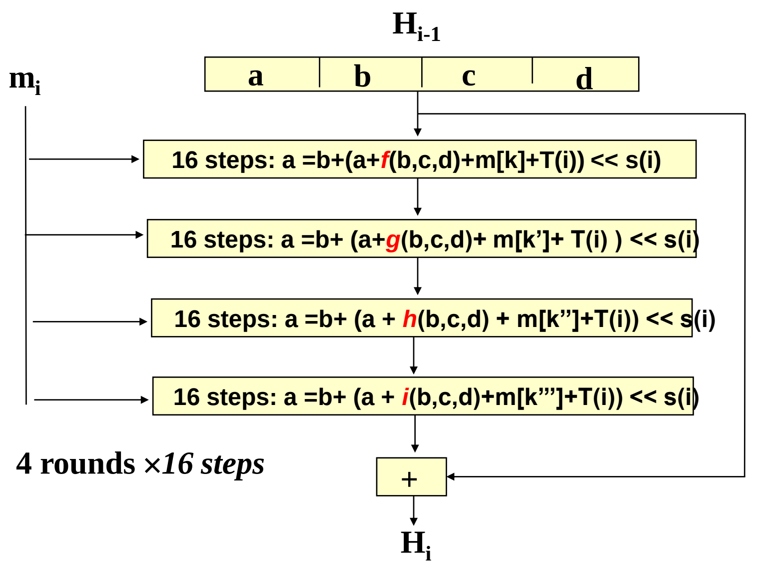
19

MD5 Overview



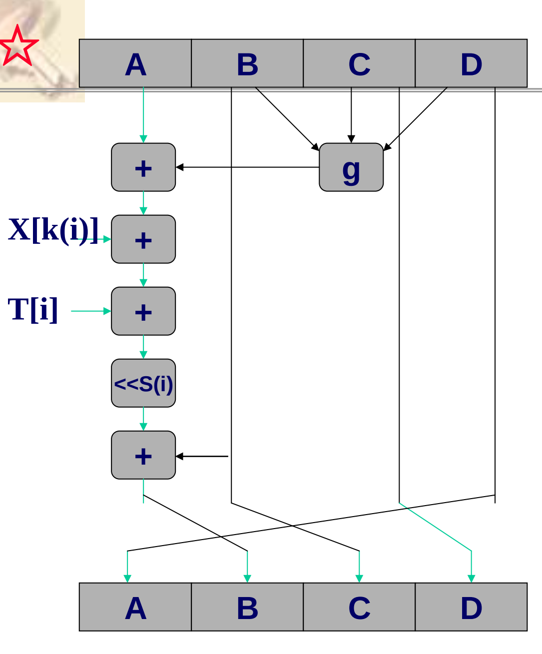
20

md5 compress function



21

md5 step



Function g(b,c,d)

1. **f(X,Y,Z) = XY v not(X) Z**
2. **g(X,Y,Z) = XZ v Y not(Z)**
3. **h(X,Y,Z) = X xor Y xor Z**
4. **i(X,Y,Z) = Y xor (X v not(Z))**

$k_1(i) = i$

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

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

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

22

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

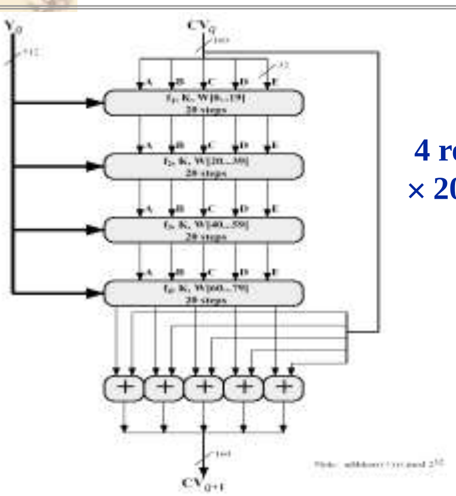
W(t)=M(t) t=0,…15;   *for t=16,…79:*

$W(t)=rot^1($ w(t-16) ⊕ w(t-3) ⊕ w(t-8) ⊕ w(t-14) $)$

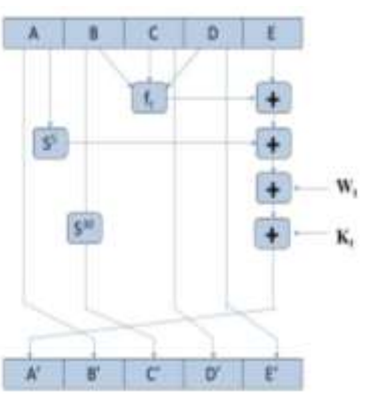
- modification in rotation (rot^1: from SHA-0)
- Same padding as MD4
- RFC3174, FIPS 180-1 (1995)

23

SHA-1 compress function



4 rounds
× 20 steps



SHA-1 step

24

SHA-1 step functions

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

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

  $B \leftarrow B<<30$

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

  - $f(t,B,C,D) = (BC) \oplus (\neg BD)$ (ch) $0 \le t < 20$
    $K[t] = 2^{30} \times sqrt(2)$
  - $h(t,B,C,D) = B \oplus C \oplus D$ (parity) $20 \le t < 40$
    $K[t] = 2^{30} \times sqrt(3)$
  - $g(t,B,C,D) = (BC) \oplus (BD) \oplus (CD)$ (maj) $30 \le t < 60$
    $K[t] = 2^{30} \times sqrt(5)$
  - $h(t,B,C,D) = B \oplus C \oplus D$ $60 \le t < 80$
    $K[t] = 2^{30} \times sqrt(10)$

25

SHA-224, 256, 384, 512

SHA	length	Message length	unit	IV	Message block	constants	Steps
-1	160 =5x32	$<2^{64}$	32-bit	0123…	512	5a827999 6ed9eba1 8f1bbcdc ca62c1d6	4X20
-256	256 =8x32	$<2^{64}$	32-bit	Sqrt(p_i) i=1-8	512	$P_i^{1/3}$ i=1-64	64
-224	224 Truncate SHA-256	$<2^{64}$	32-bit	Sqrt(p_i) i=1-8	512	$P_i^{1/3}$ i=1-64	64
-384	384 Truncate SHA-512	$<2^{128}$	64-bit	Sqrt(p_i) i=1-8	1024	$P_i^{1/3}$ i=1-80	80
-512	512 =8x64	$<2^{128}$	64-bit	Sqrt(p_i) i=1-8	1024	$P_i^{1/3}$ i=1-80	80

FIPS 180-2 [NIST 2002]

26

SHA-256 functions

Non-linear functions used:

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

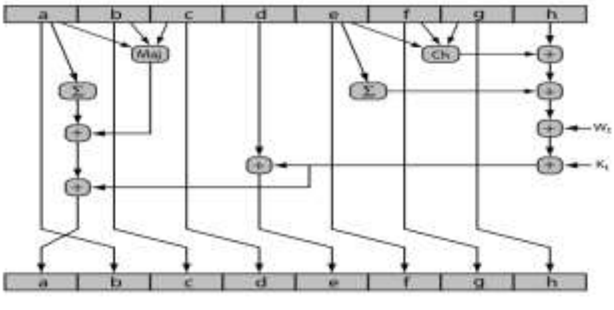
$W(t) = M(t) \quad t = 0,\dots,15$

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

Operations on 32-bit words

27

SHA-256



8-block Feistel structure

+ mod 2^{32}

- **(a,b,c,d,e,f,g,h)=(h0,h1,h2,h3,h4,h5,h6,h7)**
- **64 steps:** $0 \le t < 64$

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

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

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

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

- **(h0,h1,h2,h3,h4,h5,h6,h7)=(a,b,c,d,e,f,g,h)+(h0,h1,h2,h3,h4,h5,h6,h7)** **DM**

28

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

  $(a,b,c,d,e)_{i-1};(m_0,m_1,\ldots m_{15})_i \text{ -> } (a,b,c,d,e)_i$

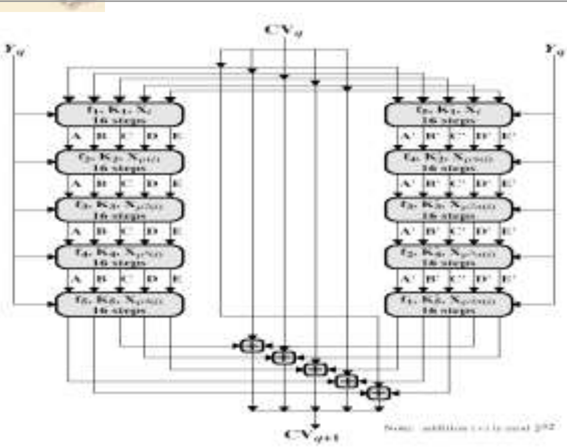
  – "Parallel 5-block MD5"

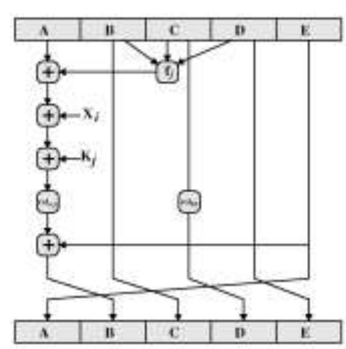
  – 160-bit hash code, comparable with SHA-1

- RipeMD-128: "Parallel MD5"
- RipeMD: "Parallel MD4"

29

RIPEMD-160



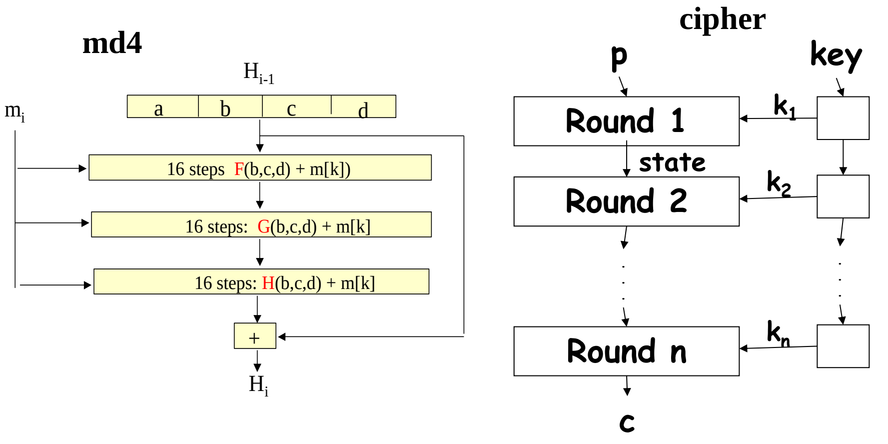


Parallel: 5 rounds X16 steps // 5 rounds X16 steps

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

30

Attack on Hash vs attack on 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

31

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 ~2^6
  - Target (2nd pre-image) attack with success probability 2^{-56}
- Pre-image: [Zhong-Lai,2011] Complexity 2^{95}

32

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
  $$md5(H_0,M) = md5(H_1, M)$$
- Dobbertin [96]: semi free-start collisions (different messages, chosen IV) on compress function: Find H, M, $M' \neq M$, but
  $$md5(H,M) = md5(H, M')$$
- Wang et.al [Crypto 04, Eurocrypt 05]: collision attack with complexity 2^{37} $(2^{39},2^{32})$
  $$MD5(H_0,M) = MD5(H_0,M')$$

33

Broken hash ?

What is "broken"?

- Academically broken: attacks with complexity less than brute-force;
- Practically broken: user or vender 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

34

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

35

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

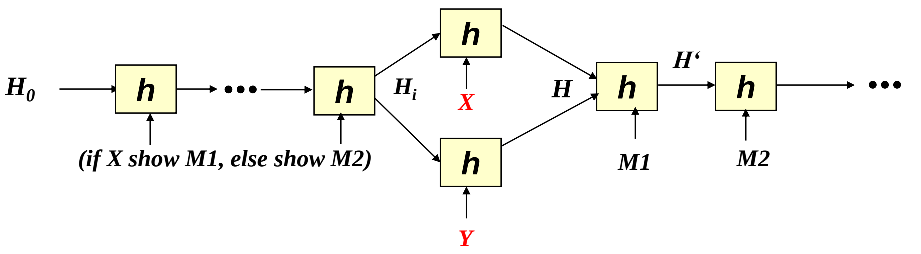
M2: an order letter for Alice's privilege

- Both letters have the same signature because of MD5(M1)=MD5(M2)

- The Boss will sign M1 (harmless)
- Alice can then use M2

36

Document collision with MD5

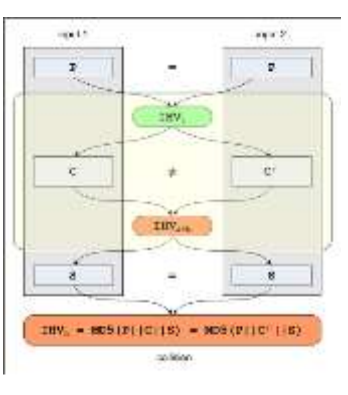


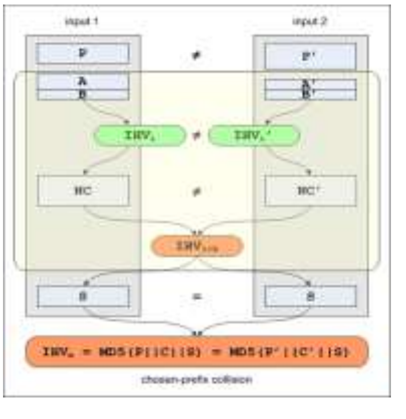
- Fixed H_0, select prefix message, from the resulting H_i, find colliding messages X, Y; the attach M1 and M2.
- (instruction,X,M1,M2)
- (instruction,Y,M1,M2)
- Have same hash code (signature)

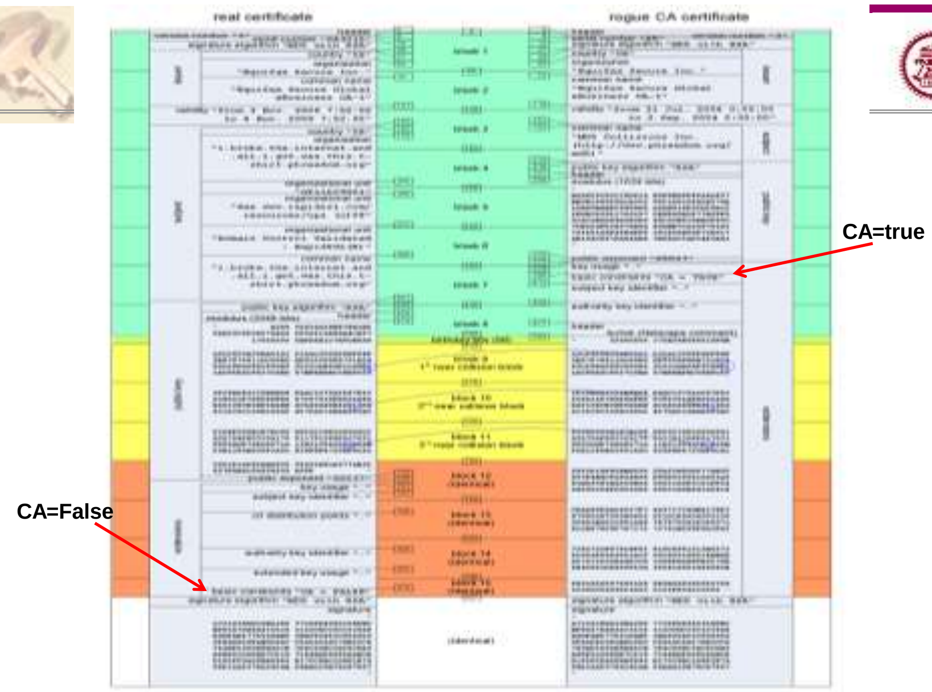
37

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}







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

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

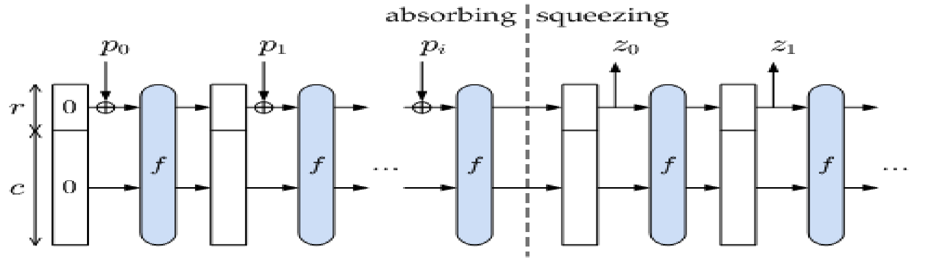
SHA-3 candidates

	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

43

Sponge Construction



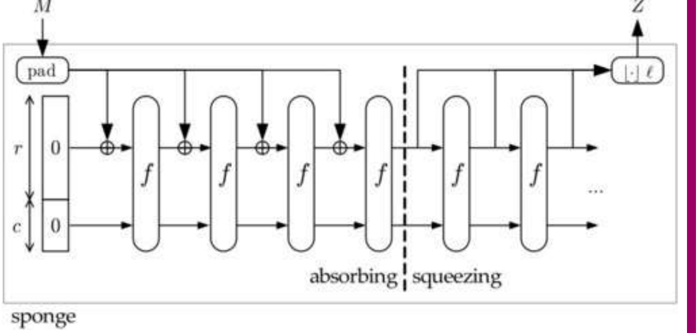
The sponge construction (larger)

- $(p_0,\ldots p_i)$ input (message)
- $(z_0,z_1,..)$ 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=2XHash



- 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

45

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/permutation. [Bertoni08]

SHA-3: Keccak

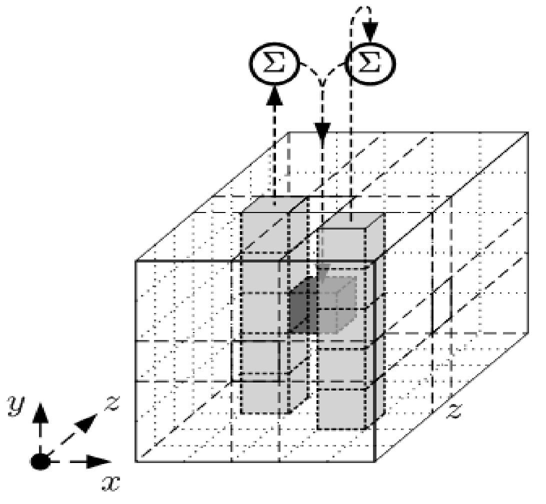
KECCAK-$f[b]$ is an iterated permutation, consisting of a sequence of n_r rounds R, indexed with i_r from 0 to $n_r - 1$. A round consists of five steps:

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

$$\begin{aligned}
\theta: \quad a[x][y][z] &\leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1], \\
\rho: \quad a[x][y][z] &\leftarrow a[x][y][z-(t+1)(t+2)/2], \\
&\quad \text{with } t \text{ satisfying } 0 \le t < 24 \text{ and } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in } \mathrm{GF}(5)^{2\times 2}, \\
&\quad \text{or } t = -1 \text{ if } x = y = 0, \\
\pi: \quad a[x][y] &\leftarrow a[x'][y'], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}, \\
\chi: \quad a[x] &\leftarrow a[x] + (a[x+1]+1)a[x+2], \\
\iota: \quad a &\leftarrow a + \mathrm{RC}[i_r].
\end{aligned}$$

SHA-3: b=1600 bits = 64 slices of 5 × 5 bits

Function θ



48

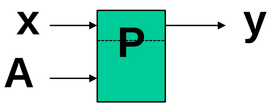
DoP: One-way functions

Besides D-log

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

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

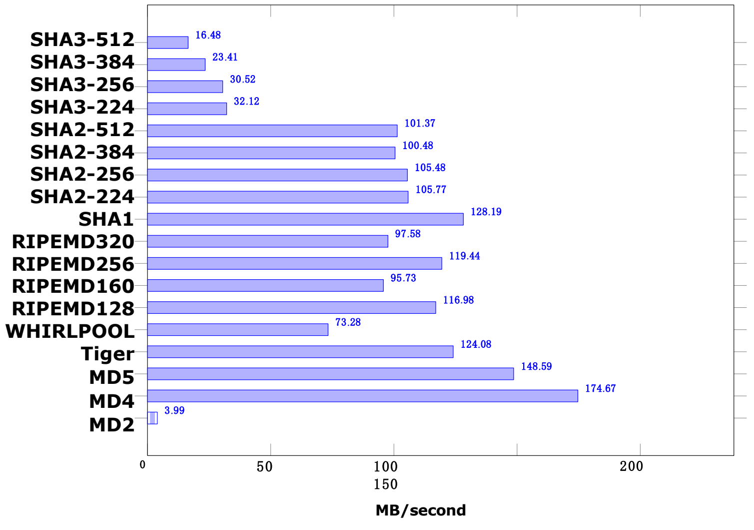
3.Permutation P(x,X)=(y,Y), y=f(x)=P^t(x,A) is one-way



“more art than science” -- *art then science*

50

HASH Performance



51

One-way functions

- Oneway function f: X ->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.
  - 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: $y=x^e$ mod N, T: N=p*q

52

MAC: Message authentication Code

- a MAC is a cryptographic checksum
  $MAC = C_K(M)$
  - 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

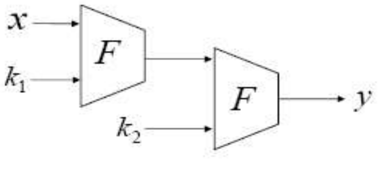
- HMAC (Internet standard RFC2104)
  - uses hash function on the message:

    $HMAC_K$ = Hash[(K^+ XOR opad) || Hash[(K^+ XOR ipad)||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(k2,H(k1,x))
- **Essentially, Hash(Key|Message|Key)**

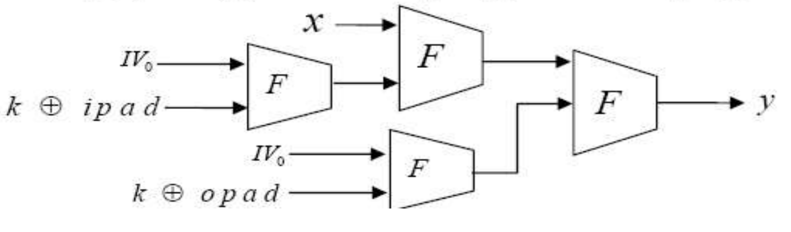
NMAC and HMAC

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

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



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



Exercise 10

1. What would happen if we use an m-bit block cipher directly (i.e. without DM) as compress function $H_i = h(H_{i-1}, M_i) = e_{M_i}(H_{i-1})$ **?**
   – estimate the complexity of target attack with and without free-start.

2. Can we use a MAC to provide non-repudiation, and why?

Deadline: before next lecture

Format: Subject: CS381-某某某-EX.#