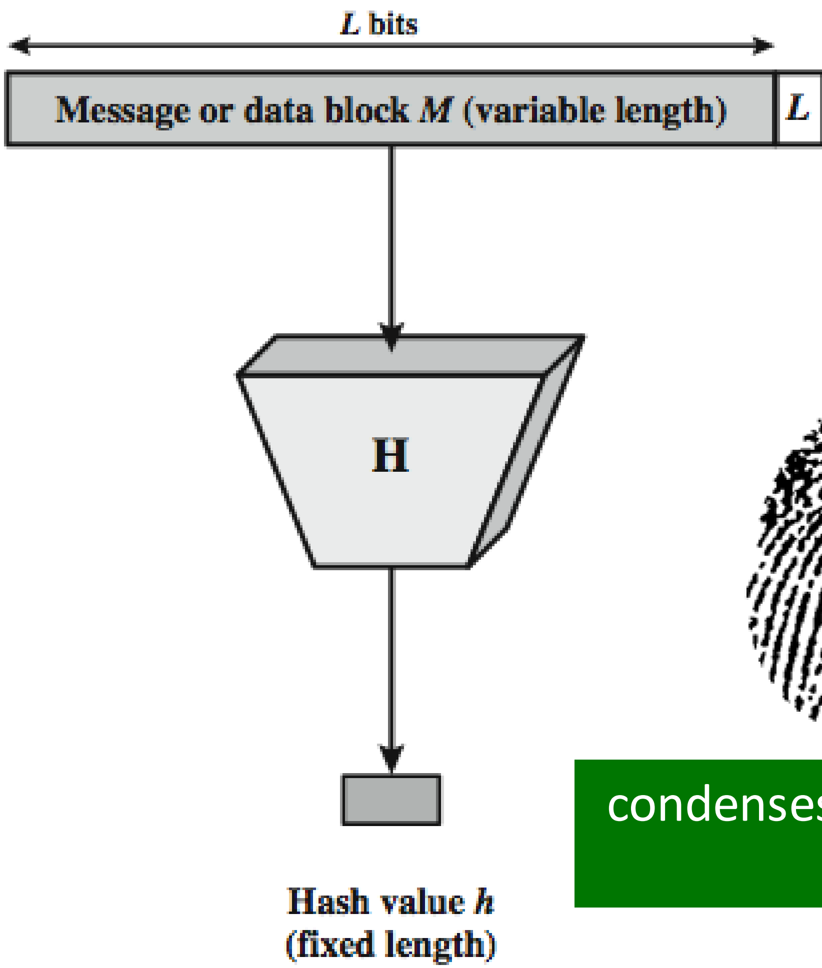
Cryptographic Hash Functions

by M.K.Chavan

Topics

- **Overview of Cryptography Hash Function**
- Usages
- Properties
- Hashing Function Structure
- Attack on Hash Function
- The Road to new Secure Hash Standard

Hash Function

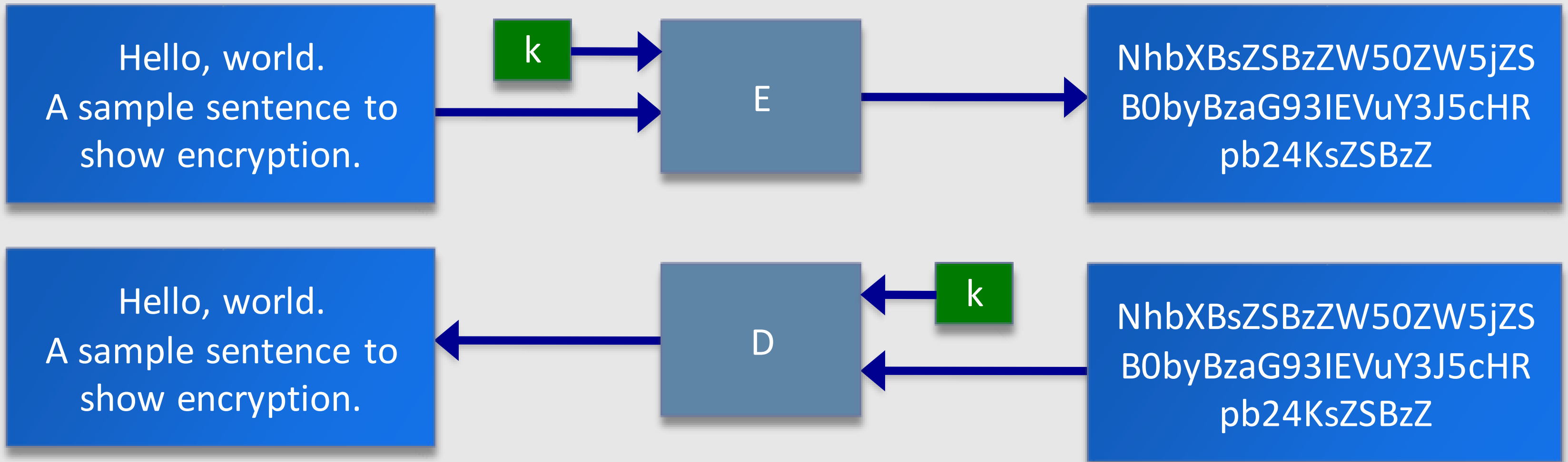


- The hash value represents concisely the longer message
  - may called the *message digest*
- A message digest is as a "digital fingerprint'' of the original document

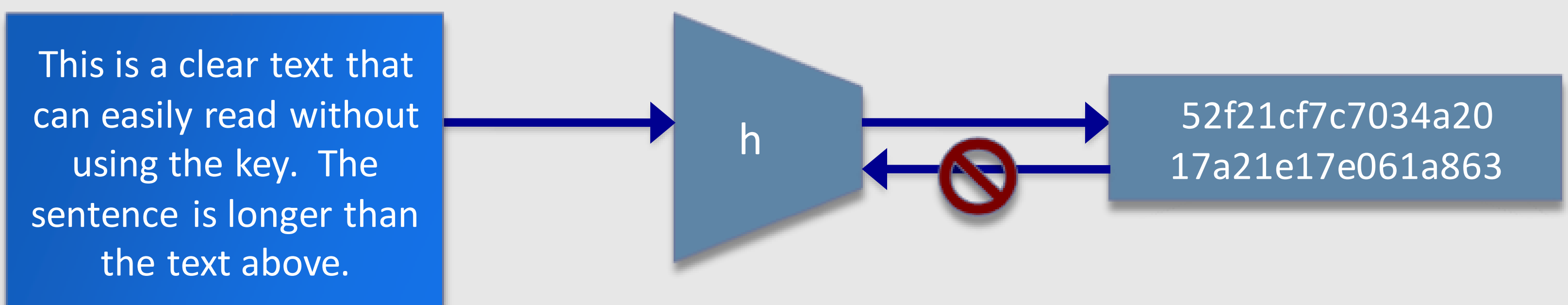
condenses arbitrary message to fixed size

$$h = H(M)$$

Hashing V.S. Encryption



- Encryption is two way, and requires a key to encrypt/decrypt



- Hashing is one-way. There is no 'de-hashing’

Motivation for Hash Algorithms

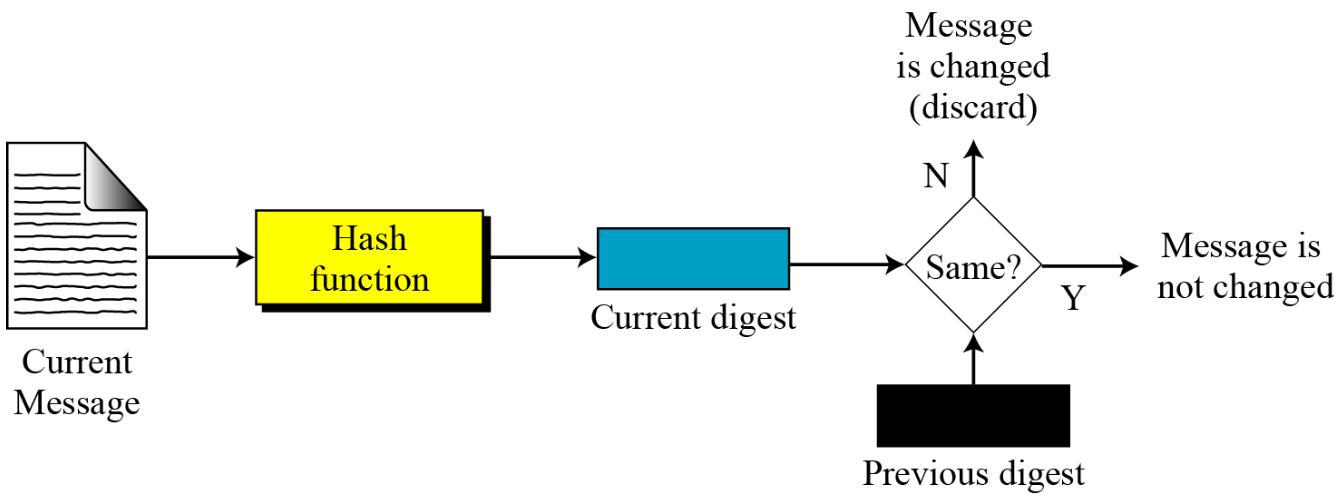
- Goal
  - Design a code where the original message can not be inferred based on its checksum
  - such that an accidental or intentional change to the message will change the hash value

Hash Function Applications

- Used Alone
  - Fingerprint -- file integrity verification
  - Password storage (one-way encryption)

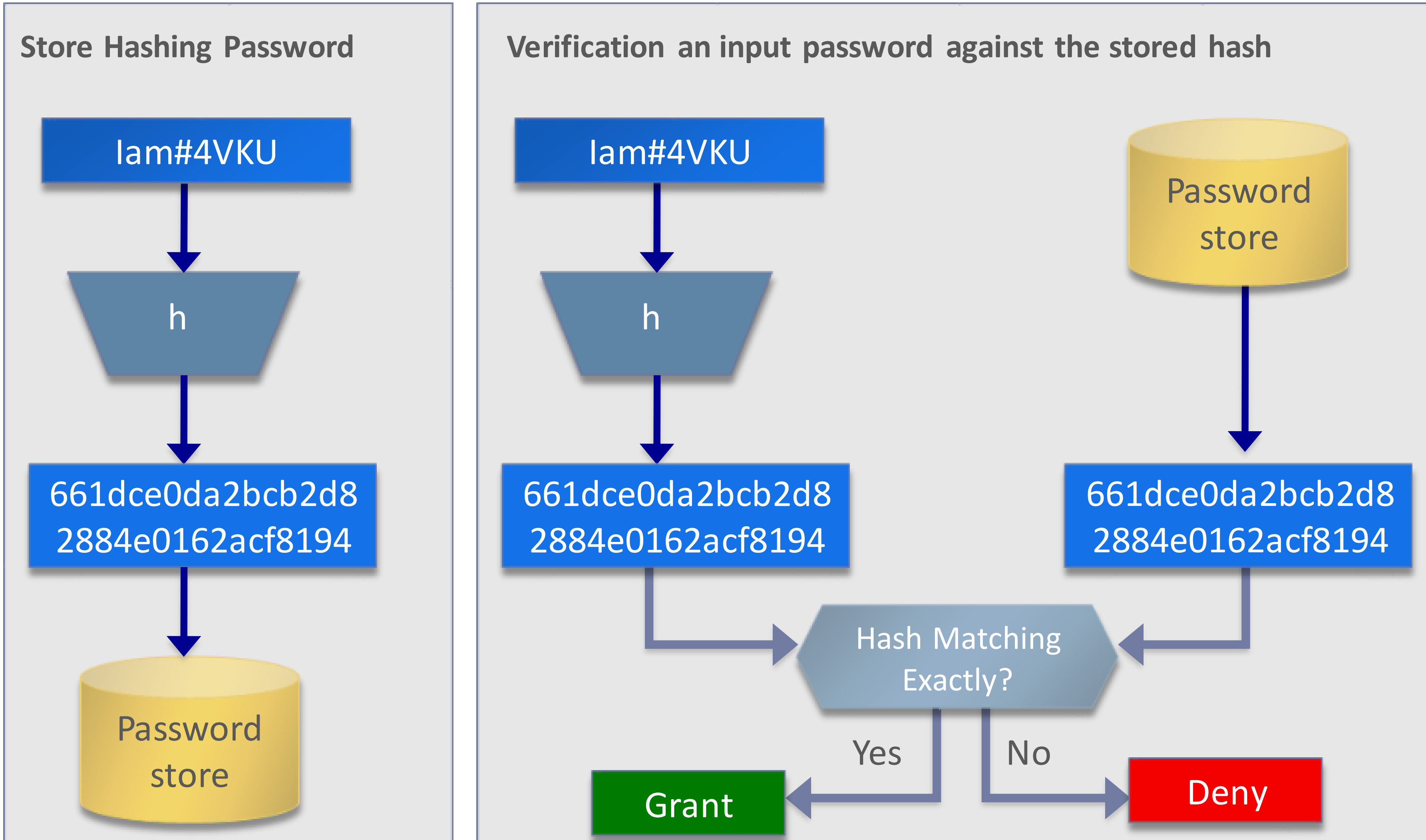
- Combined with encryption functions
  - Hash based Message Authentication Code (HMAC)
    - protects both a message's integrity and confidentiality
  - Digital signature
    - Ensuring Non-repudiation
    - Encrypt hash with private (signing) key and verify with public (verification) key

Integrity



- to create a one-way password file
  - store hash of password not actual password
- for intrusion detection and virus detection
  - keep & check hash of files on system

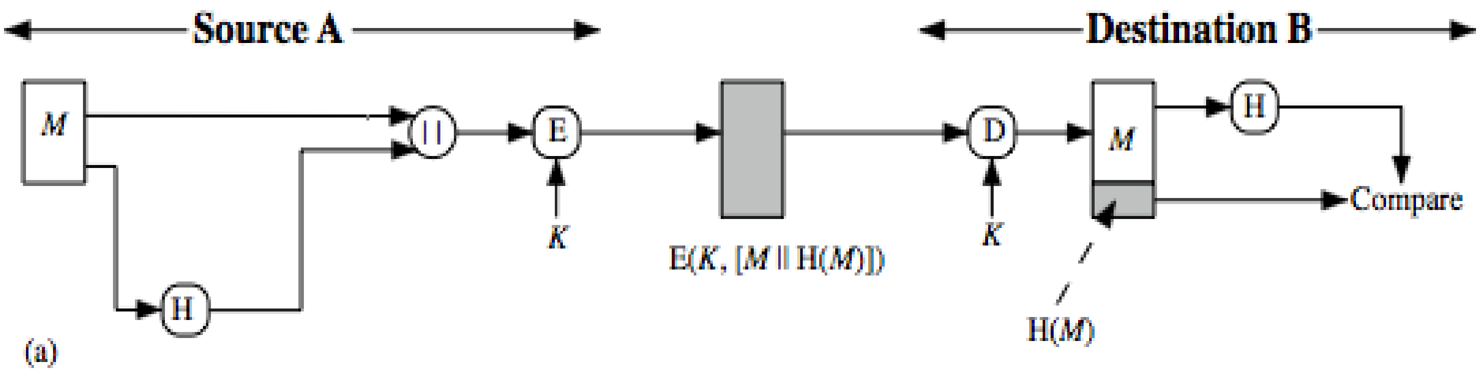
Password Verification



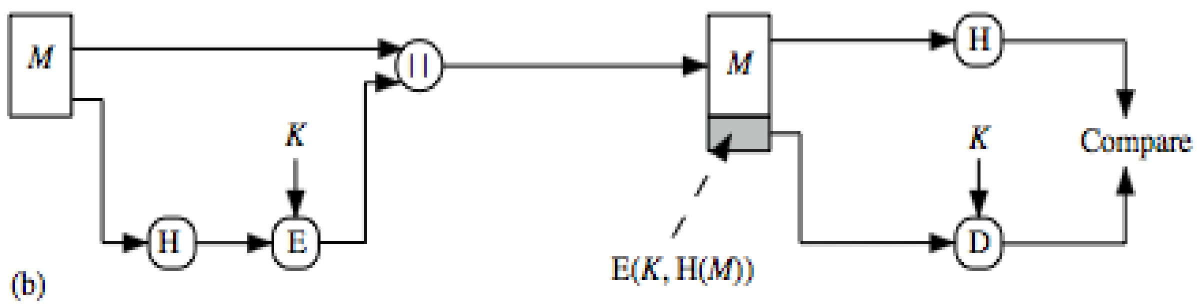
Topics

- Overview of Cryptography Hash Function
- **Usages**
- Properties
- Hashing Function Structure
- Attack on Hash Function
- The Road to new Secure Hash Standard

Hash Function Usages (I)

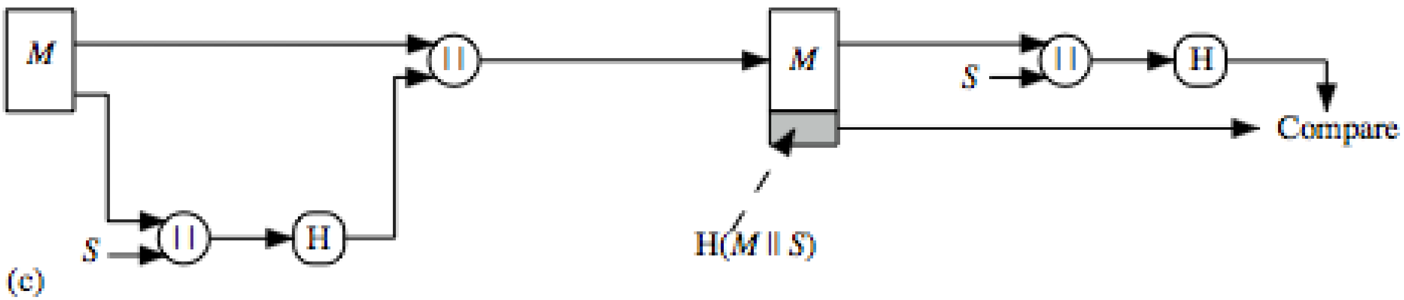


Message encrypted : Confidentiality and authentication

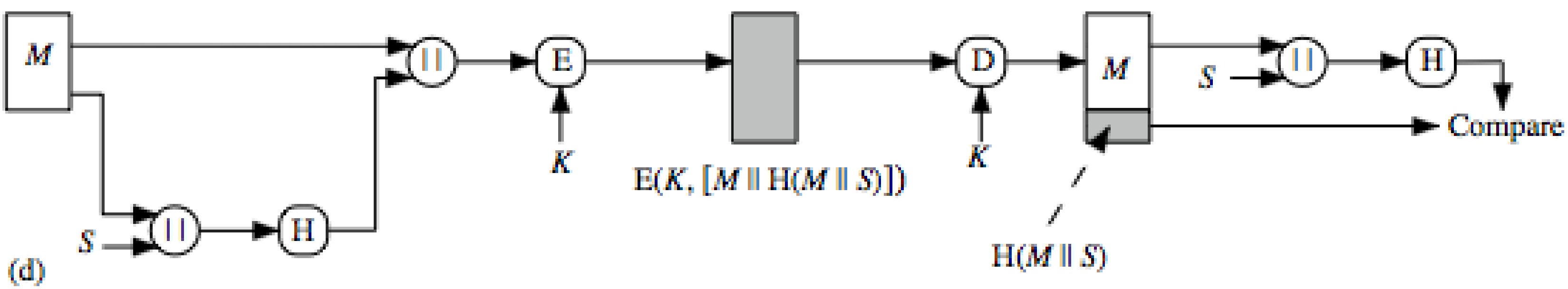


Message unencrypted: Authentication

Hash Function Usages (II)

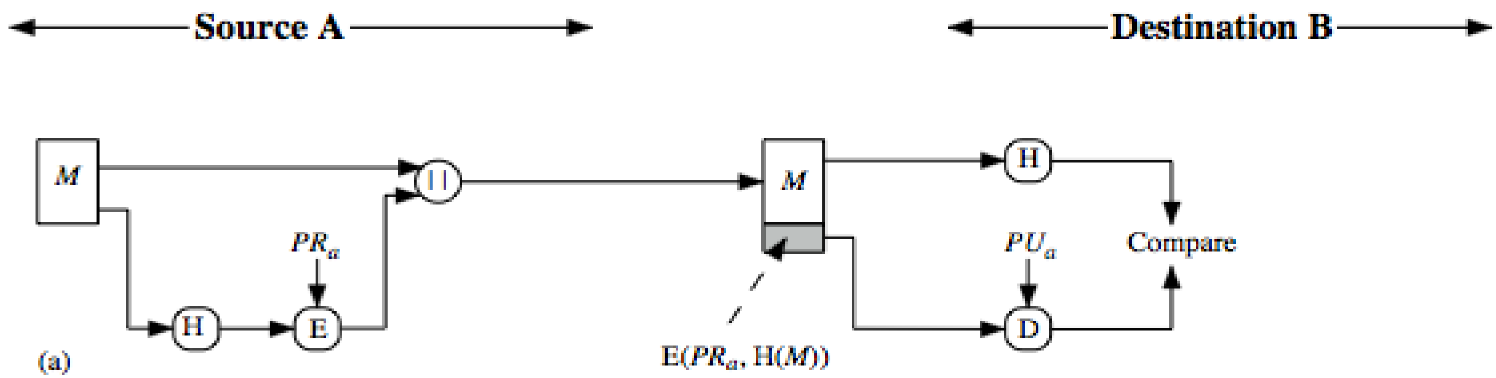


Message encrypted : Authentication (no encryption needed!)

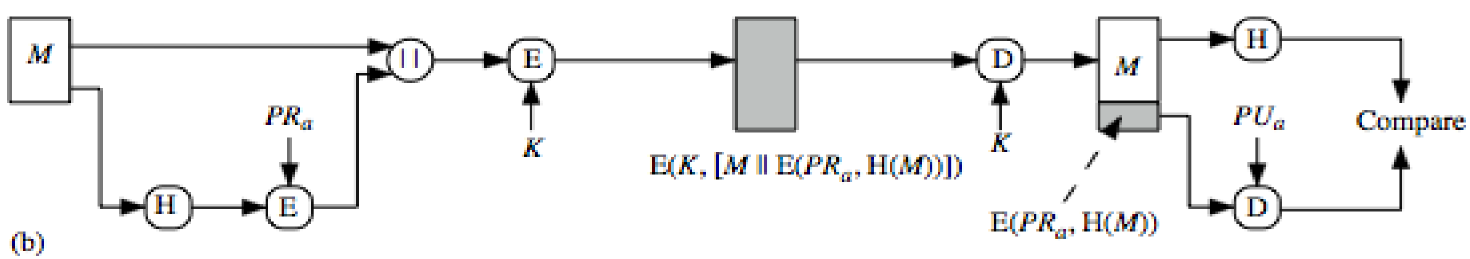


Message unencrypted: Authentication, confidentiality

Hash Function Usages (III)



Authentication, digital signature



Authentication, digital signature, confidentiality

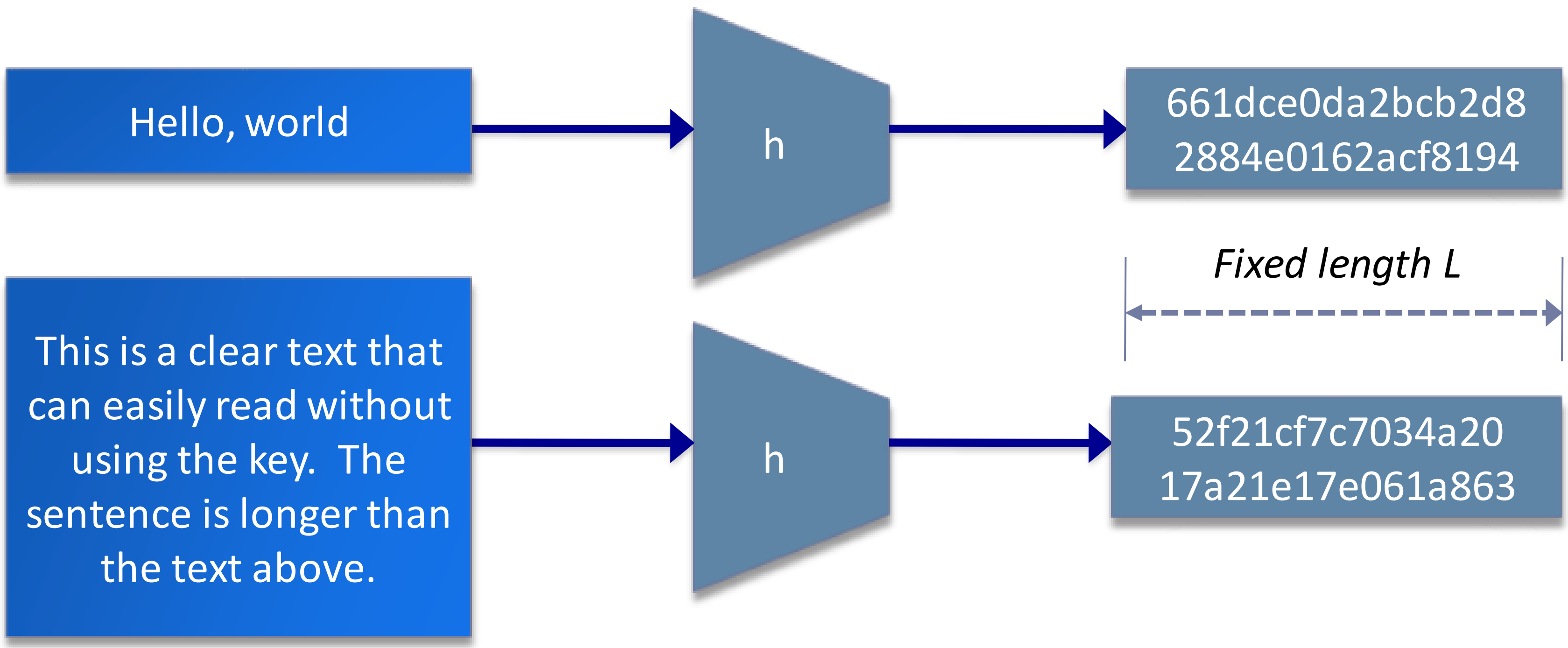
Topics

- Overview of Cryptography Hash Function
- Usages
- **Properties**
- Hashing Function Structure
- Attack on Hash Function
- The Road to new Secure Hash Standard

Hash Function Properties

- Arbitrary-length message to fixed-length digest
- Preimage resistant (**One-way property**)
- Collision resistant (**Strong collision resistance**)

Properties : Fixed length



- Arbitrary-length message to fixed-length digest

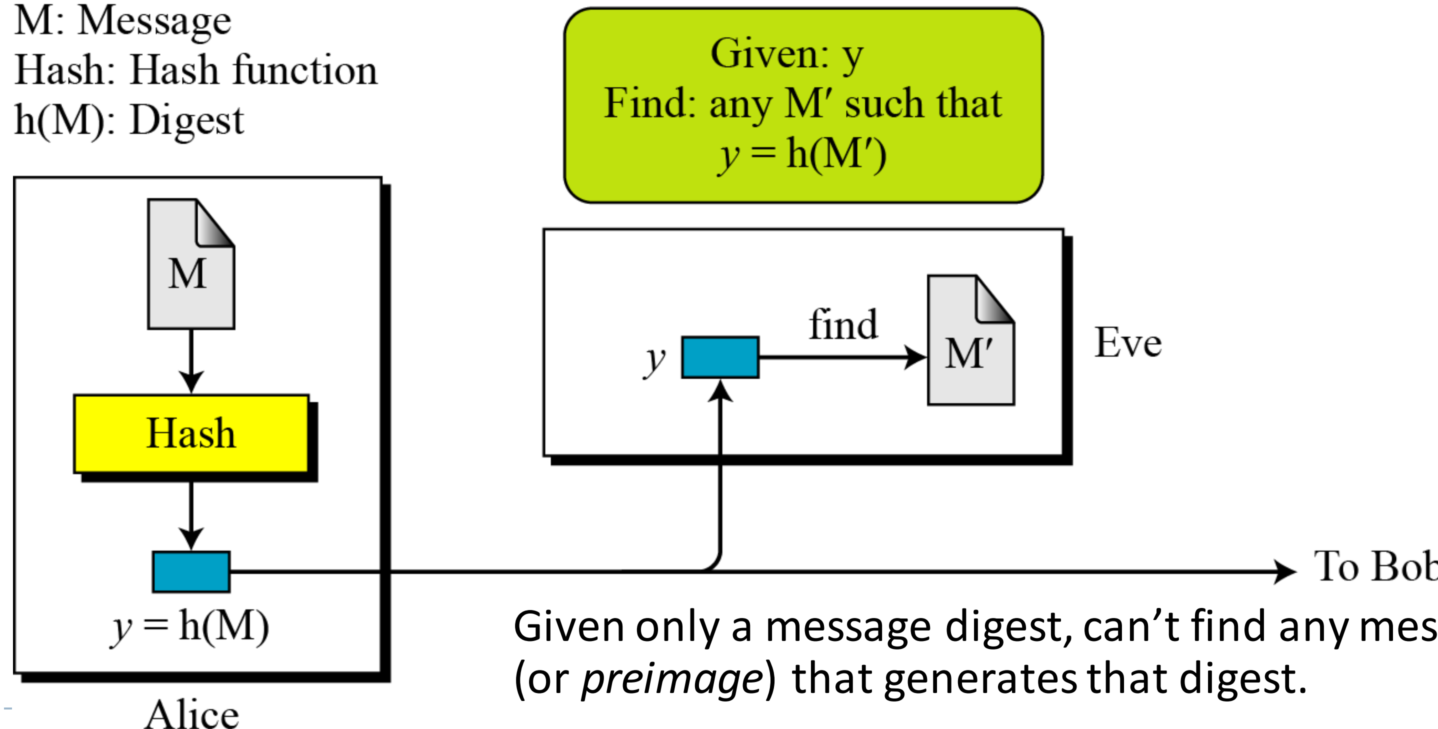
Preimage resistant

- This measures how difficult to devise a message which hashes to the known digest
- Roughly speaking, the hash function must be one-way.

Preimage Attack

Given: $y = h(M)$ **Find:** M' such that $y = h(M')$

M: Message
Hash: Hash function
h(M): Digest



Given only a message digest, can't find any message (or *preimage*) that generates that digest.

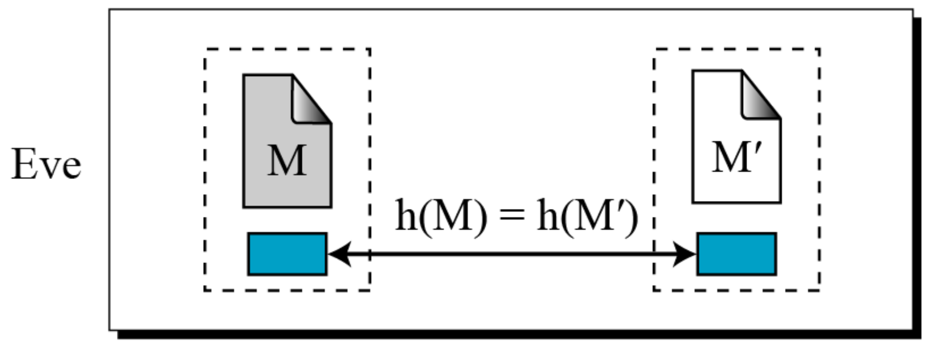
Collision Resistant

Collision Attack

Given: none | **Find: $M' \neq M$ such that $h(M) = h(M')$**

M: Message
Hash: Hash function
h(M): Digest

Find: M and M′ such that $M \neq M'$, but $h(M) = h(M')$



- Can't find any two different messages with the same message digest
  - Collision resistance implies second preimage resistance
  - Collisions, if we could find them, would give signatories a way to repudiate their signatures

Topics

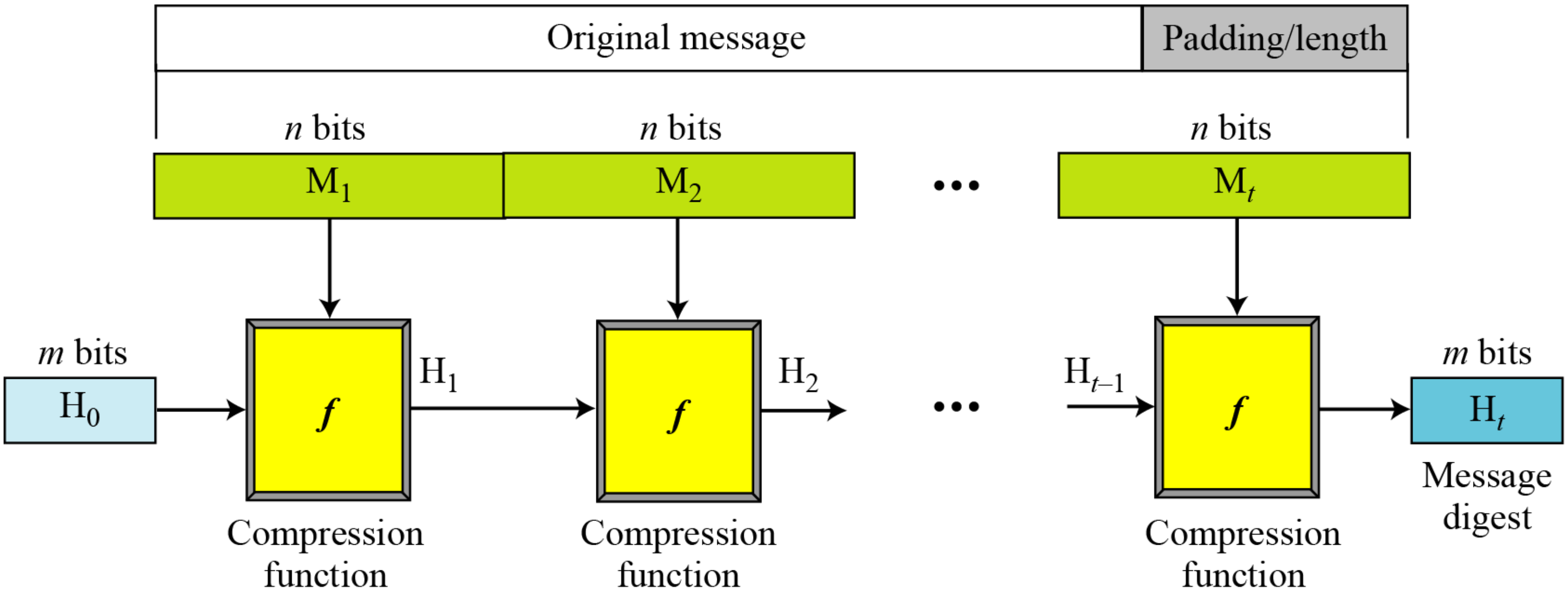
- Overview of Cryptography Hash Function
- Usages
- Properties
- **Hashing Function Structure**
- Attack on Hash Function
- The Road to new Secure Hash Standard

Two Group of Compression Functions

- The compression function is made from scratch
  - Message Digest

- A symmetric-key block cipher serves as a compression function
  - Whirlpool

Merkle-Damgard Scheme



- Well-known method to build cryptographic hash function
- A message of arbitrary length is broken into blocks
  - length depends on the compression function *f*
  - padding the size of the message into a multiple of the block size.
  - sequentially process blocks , taking as input the result of the hash so far and the current message block, with the final fixed length output

Hash Functions Family

- **MD (Message Digest)**
  - Designed by Ron Rivest
  - Family: MD2, MD4, MD5
- **SHA (Secure Hash Algorithm)**
  - Designed by NIST
  - Family: SHA-0, SHA-1, and SHA-2
    - SHA-2: SHA-224, SHA-256, SHA-384, SHA-512
    - SHA-3: New standard in competition
- **RIPEMD (Race Integrity Primitive Evaluation Message Digest)**
  - Developed by Katholieke University Leuven Team
  - Family : RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320

MD5, SHA-1, and RIPEMD-160

	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	2^{64} - 1 bits	2^{64} - 1 bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

MD2, MD4 and MD5

- Family of one-way hash functions by Ronald Rivest
  - All produces 128 bits hash value
- **MD2: 1989**
  - Optimized for 8 bit computer
  - Collision found in 1995
- **MD4: 1990**
  - Full round collision attack found in 1995
- **MD5: 1992**
  - Specified as Internet standard in RFC 1321
  - Practical Collision MD5 has been broken since 2004
  - CA attack published in 2007

Topics

- Overview of Cryptography Hash Function
- Usages
- Properties
- **Hashing Function Structure**
  - **MD5**
  - SHA
- Attack on Hash Function
- The Road to new Secure Hash Standard

Topics

- Overview of Cryptography Hash Function
- Usages
- Properties
- **Hashing Function Structure**
  - MD5
  - **SHA**
- Attack on Hash Function
- The Road to new Secure Hash Standard

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
  - revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - based on design of MD4 with key differences
- produces 160-bit hash values
- recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

Revised SHA

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

SHA Versions

	MD5	SHA-0	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Digest size	128	160	160	224	256	384	512
Message size	2^{64}-1	2^{64}-1	2^{64}-1	2^{64} -1	2^{64}-1	2^{128}-1	2^{128}-1
Block size	512	512	512	512	512	1024	1024
Word size	32	32	32	32	32	64	64
# of steps	64	64	80	64	64	80	80

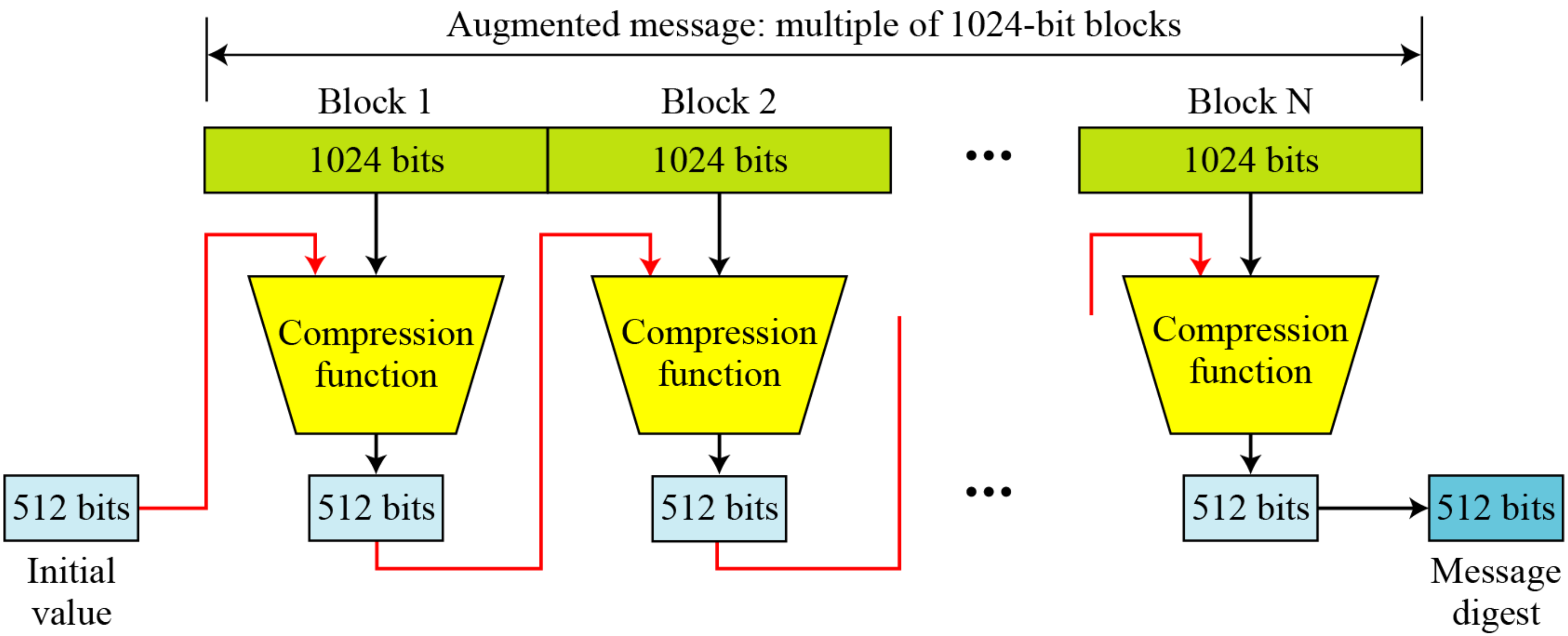
Full collision found

Sample Processing

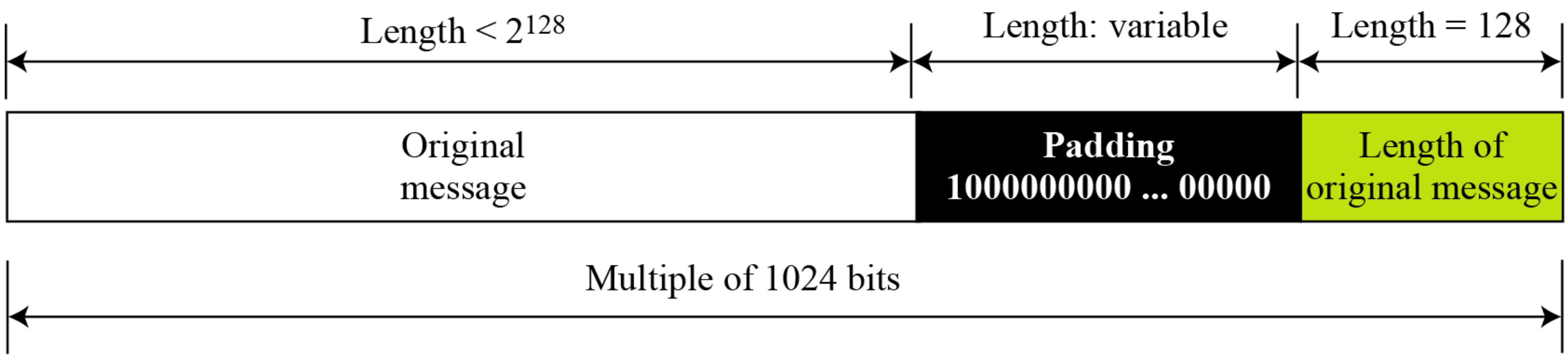
Type	bits	data processed
MD5	128	469.7 MB/s
SHA-1	160	339.4 MB/s
SHA-512	512	177.7 MB/s

- Mac Intel 2.66 Ghz core i7
- 1024 bytes block of data

SHA-512 Overview



Padding and length field in SHA-512

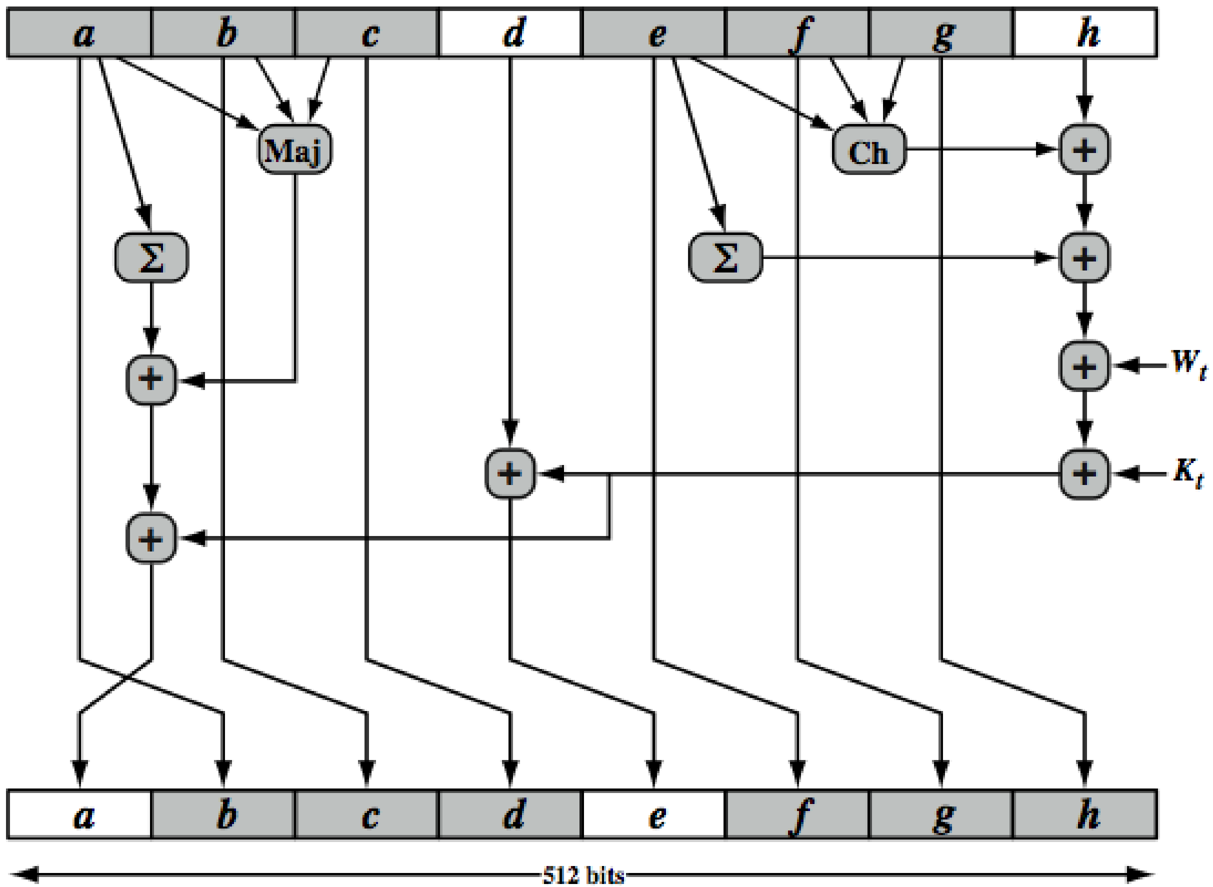


- **What is the number of padding bits if the length of the original message is 2590 bits?**
- We can calculate the number of padding bits as follows:

$$|P| = (-2590 - 128) \bmod 1024 = -2718 \bmod 1024 = 354$$

- The padding consists of one 1 followed by 353 0's.

SHA-512 Round Function

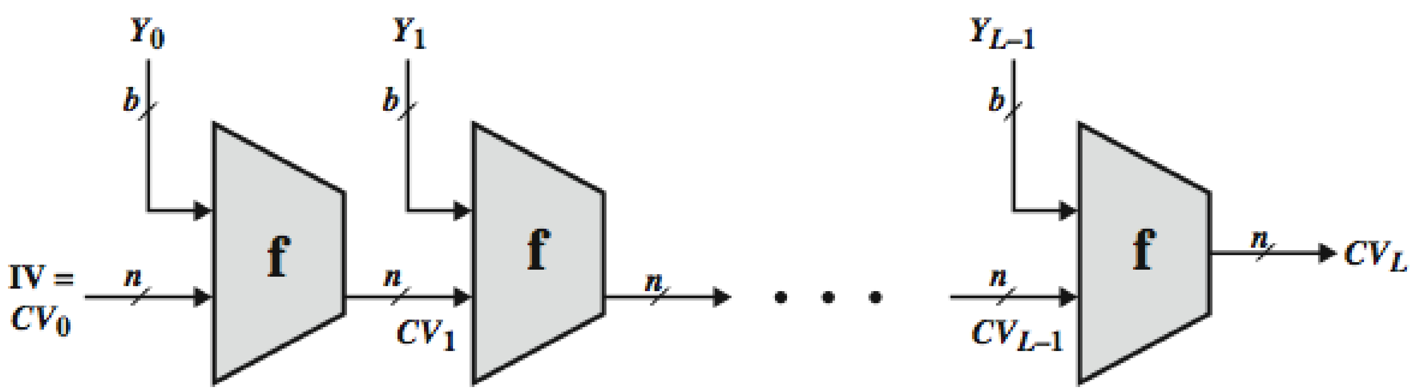


Topics

- Overview of Cryptography Hash Function
- Usages
- Properties
- Hashing Function Structure
  - MD5
  - SHA

Hash Function Cryptanalysis

- cryptanalytic attacks exploit some property of algorithm so faster than exhaustive search
- hash functions use iterative structure
  - process message in blocks (incl length)
- attacks focus on collisions in function f



Attacks on Hash Functions

- brute-force attacks and cryptanalysis
  - cryptanalytic attacks exploit some property of algorithm so faster than brute-force
- a preimage or second preimage attack
  - find y such that $H(y)$ equals a given hash value
- collision resistance
  - find two messages x & y with same hash so $H(x) = H(y)$

"*md5 and sha1 are both clearly broken (in terms of collision-resistance*"

Ron Rivest

http://mail.python.org/pipermail/python-dev/2005-December/058850.html

Topics

- Overview of Cryptography Hash Function
- Usages
- Properties
- Hashing Function Structure
  - MD5
  - SHA
- Attack on Hash Function
- **The Road to new Secure Hash Standard**

The need of new Hash standard

- *MD5 should be considered cryptographically broken and unsuitable for further use*, US CERT 2010
- In 2004, a collision for the full SHA-0 algorithm was announced

- SHA-1 not yet fully "**broken**"
  - but similar to the broken MD5 & SHA-0
  - so considered insecure and be fade out
- SHA-2 (esp. SHA-512) seems secure
  - shares same structure and mathematical operations as predecessors so have concern

SHA-3 Requirements

- NIST announced in 2007 a competition for the SHA-3 next gen hash function
- Replace SHA-2 with SHA-3 in any use
  - so use same hash sizes
- preserve the nature of SHA-2
  - so must process small blocks (512 / 1024 bits)
- evaluation criteria
  - security close to theoretical max for hash sizes
  - cost in time & memory
  - characteristics: such as flexibility & simplicity

Timeline Competition

- **Nov 2007**: Announce public competition
- **Oct 2008**: 64 Entries
- **Dec 2008**: 51 Entries as 1st Round
- **Jul 2009**: 14 Entries as 2nd Round
- **Dec 2010**: 5 Entries as 3rd Round
- **Jan 2011:** Final packages submission and enter public comments
- **2012:** *SHA-3 winner announcement (Still in progress)*

Five SHA-3 Finalists

- BLAKE
- Grøstl
- JH
- Keccak
- Skien

http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html

Secure Hash Algorithm-512 (SHA-512)

M.K.Chavan

Outlines

- SHA-512 overview
- processing of SHA-521
- word expansion
- compression function
- round function
- additive constants
- example using SHA-512
- applications
- cryptanalysis

Secure Hash Algorithm-512 (SHA-512)

Outlines

- SHA-512 overview
- processing of SHA-521
- word expansion
- compression function
- round function
- additive constants
- example using SHA-512
- applications
- cryptanalysis

Overview

-
    - developed by National Institute of Standards and Technology
    - member of SHA-2 family
    - latest version of Secure Hash Algorithm
    - based on the Merkle-Damgard scheme
    - maximum message size 2^{128}-1 bits
    - block size 1024 bits
    - message digest size 512 bits
    - number of rounds 80
    - word size 64 bits

Processing of SHA-512

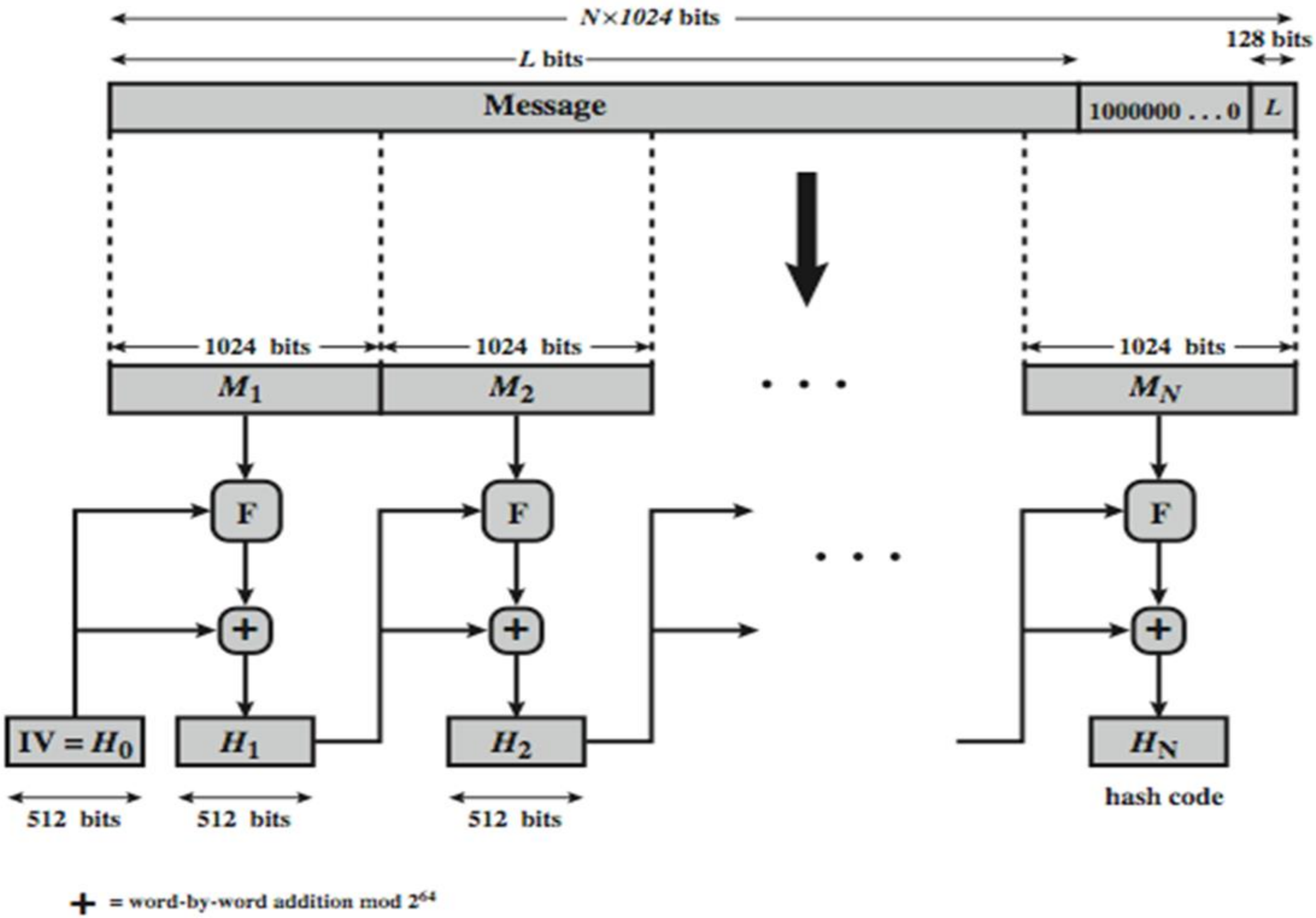


Figure: SHA-512 Processing of a Single 1024-Bit Block

Message block and digest word

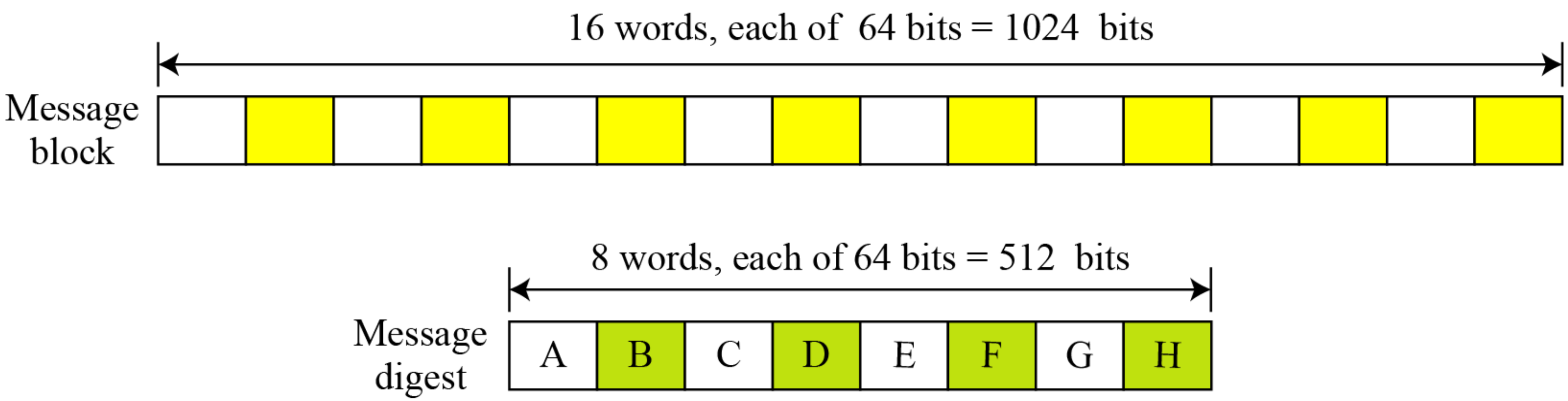


Figure: A message block and the digest as words

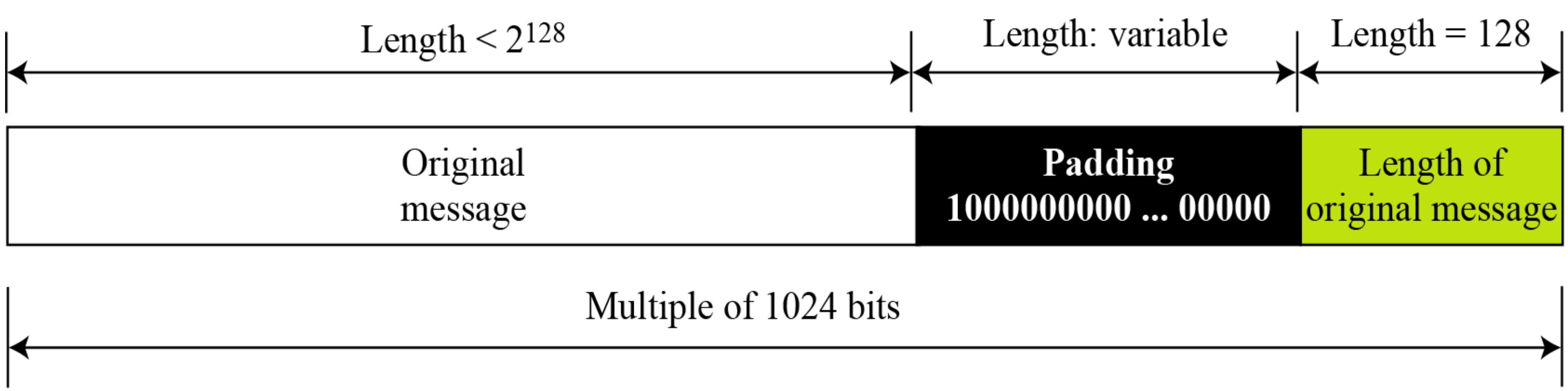


Figure: Padding and length field

Processing of SHA-512

- appending padding and fixed 128 bit length field
- dividing the augmented message into blocks
- using a 64-bit word derived from the current message block
- using 8 constants based on square root of first 8 prime numbers (2-19)
- updating a 512-bit buffer
- using a round constant based on cube root of first 80 prime numbers (2-409)

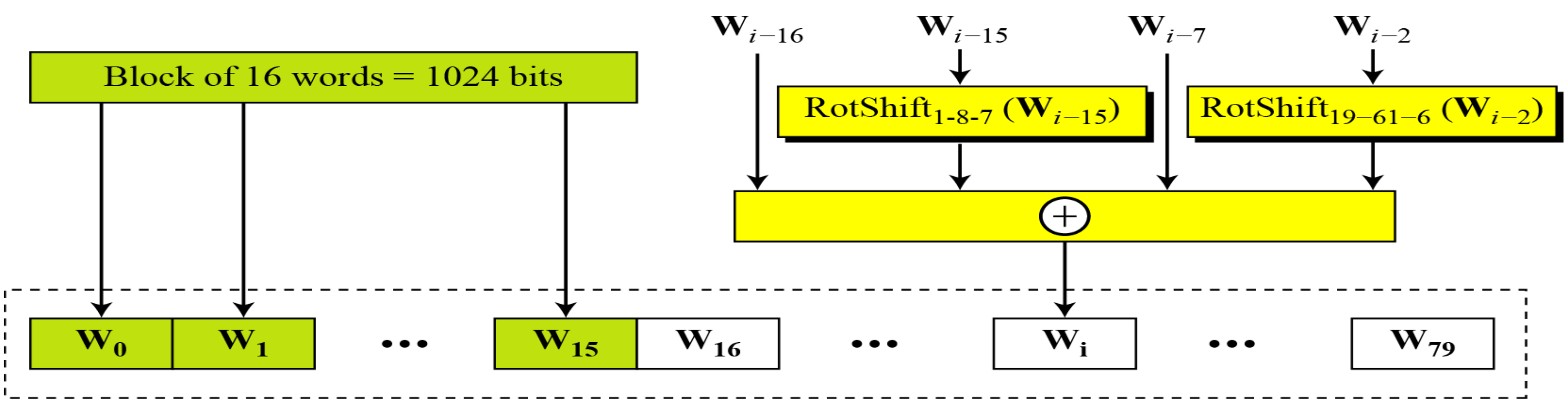
Message block and digest word

- operates on words
- each block consists of sixteen 64 bits(1024 bits) words
- message digest has eight 64 bits (512 bits) words named A,B,C,D,E,F,G,H
- expanding 80 words from sixteen 64 bits words

SHA-512 Initial Values & Word Expansion

Buffer	Value (in Hexadecimal)	Buffer	Value (in Hexadecimal)
A_0	6A09E667F3BCC908	E_0	510E527FADE682D1
B_0	BB67AE8584CAA73B	F_0	9B05688C2B3E6C1F
C_0	3C6EF372FE94F82B	G_0	1F83D9ABFB41BD6B
D_0	A54FF53A5F1D36F1	H_0	5BE0CD19137E2179

Table : Values of constants in message digest initialization of SHA-512



RotShift$_{l\text{-}m\text{-}n}$ (x): RotR$_l(x)$ $\oplus$ RotR$_m$ (x) $\oplus$ ShL$_n$ (x)

RotR$_i(x)$: Right-rotation of the argument x by i bits
ShL$_i$(x): Shift-left of the argument x by i bits and padding the left by 0's.

Figure: Word expansion in SHA-512

Calculation of constants

- For example,

The 8th prime is 19, with the square root $(19)^{1/2}$= 4.35889894354 Converting this number to binary with only 64 bits in the fraction part, we get,

$$(100.0101\ 1011\ 1110 \ldots 1001)_2 \longrightarrow (4.5BE0CD19137E2179)_{16}$$

The fraction part : $(5BE0CD19137E2179)_{16}$

The 80th prime is 409, with the cubic root $(409)^{1/3}$ = 7.42291412044. Converting this number to binary with only 64 bits in the fraction part, we get

$$(111.0110\ 1100\ 0100\ 0100 \ldots 0111)_2 \rightarrow (7.6C44198C4A475817)_{16}$$

The fraction part: $(6C44198C4A475817)_{16}$

SHA-512 Compression Function

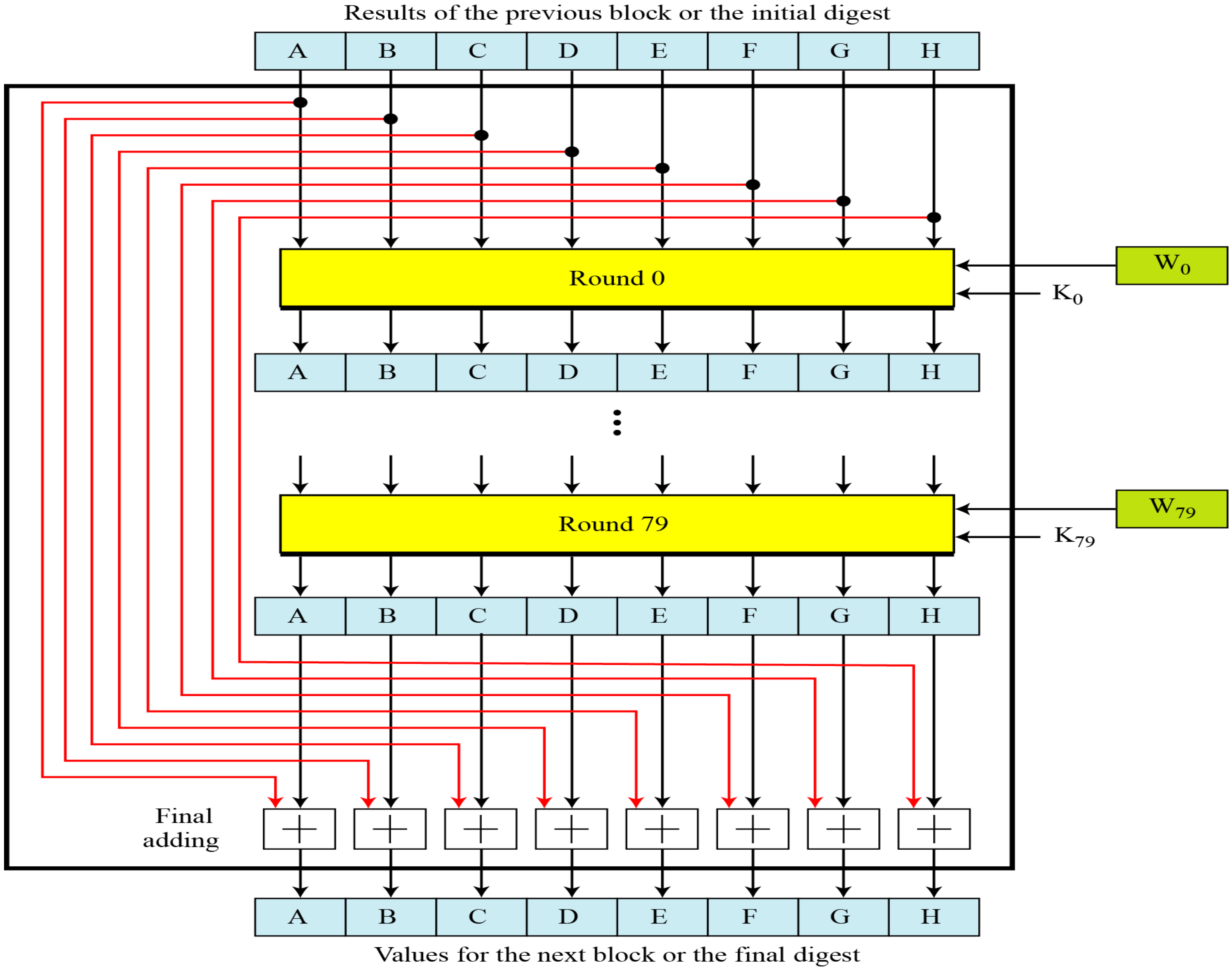


Figure: Compression Function in SHA-512

SHA-512 Round constants (K)

428A2F98D728AE22	7137449123EF65CD	B5C0FBCFEC4D3B2F	E9B5DBA58189DBBC
3956C25BF348B538	59F111F1B605D019	923F82A4AF194F9B	AB1C5ED5DA6D8118
D807AA98A3030242	12835B0145706FBE	243185BE4EE4B28C	550C7DC3D5FFB4E2
72BE5D74F27B896F	80DEB1FE3B1696B1	9BDC06A725C71235	C19BF174CF692694
E49B69C19EF14AD2	EFBE4786384F25E3	0FC19DC68B8CD5B5	240CA1CC77AC9C65
2DE92C6F592B0275	4A7484AA6EA6E483	5CB0A9DCBD41FBD4	76F988DA831153B5
983E5152EE66DFAB	A831C66D2DB43210	B00327C898FB213F	BF597FC7BEEF0EE4
C6E00BF33DA88FC2	D5A79147930AA725	06CA6351E003826F	142929670A0E6E70
27B70A8546D22FFC	2E1B21385C26C926	4D2C6DFC5AC42AED	53380D139D95B3DF
650A73548BAF63DE	766A0ABB3C77B2A8	81C2C92E47EDAEE6	92722C851482353B
A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	4597F299CFC657E2	5FCB6FAB3AD6FAEC	6C44198C4A475817

Figure: List of round constants used in SHA-512

SHA-512 Round Function

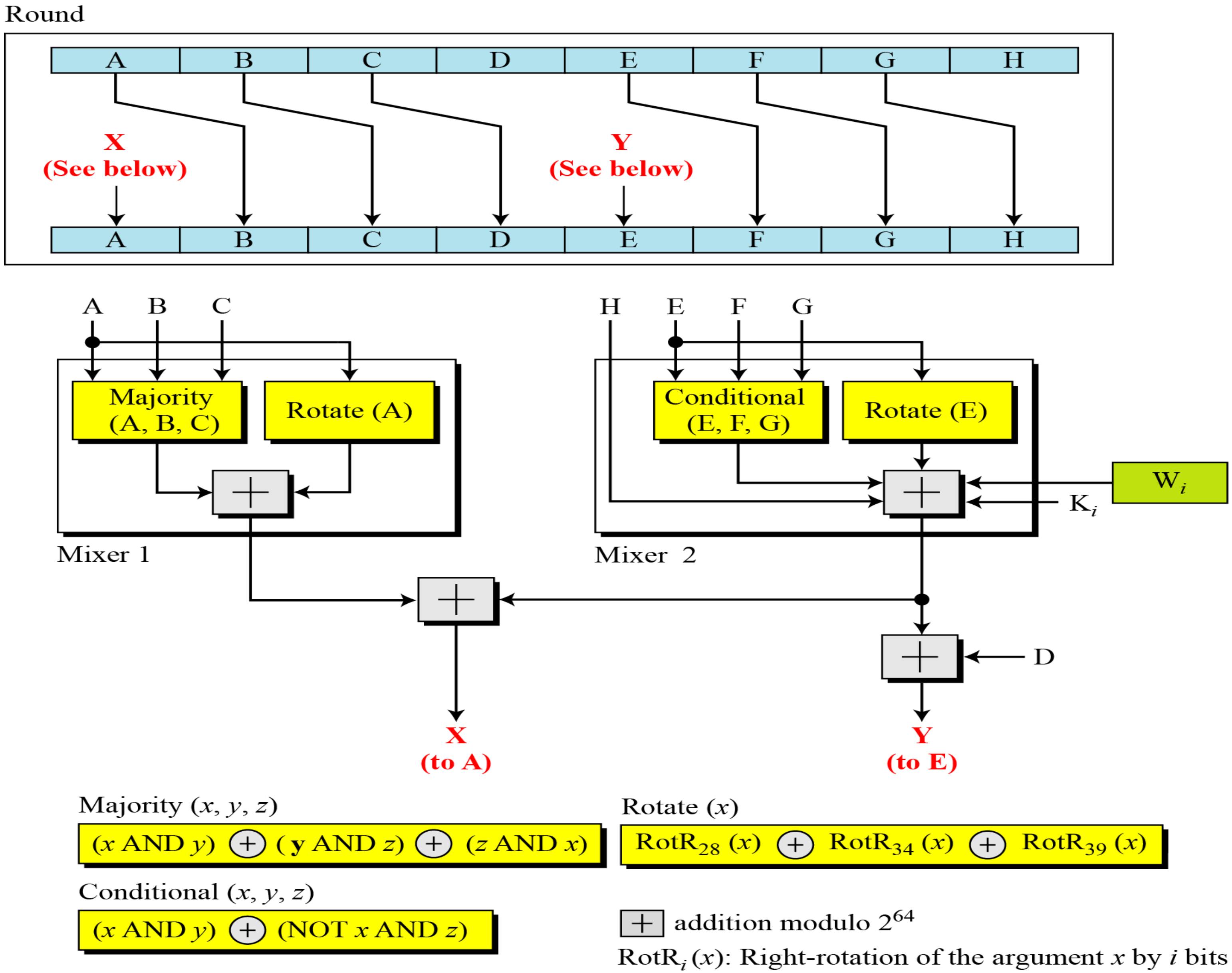


Figure: Structure of each round in SHA-512

SHA-512 Round Function

Majority Function

$$(A_j \text{ AND } B_j) \oplus (B_j \text{ AND } C_j) \oplus (C_j \text{ AND } A_j)$$

Conditional Function

$$(E_j \text{ AND } F_j) \oplus (\text{NOT } E_j \text{ AND } G_j)$$

Rotate Functions

$$\text{Rotate (A): } RotR_{28}(A) \oplus RotR_{34}(A) \oplus RotR_{29}(A)$$

$$\text{Rotate (E): } RotR_{28}(E) \oplus RotR_{34}(E) \oplus RotR_{29}(E)$$

SHA-512 Majority Function calculation

- We apply the Majority function on buffers A, B, and C. If the leftmost hexadecimal digits of these buffers are 0x7, 0xA, and 0xE, respectively, what is the leftmost digit of the result?

Solution

$$(0 \text{ AND } 1) \oplus (1 \text{ AND } 1) \oplus (1 \text{ AND } 0) = 0 \oplus 1 \oplus 0 = 1$$

The digits in binary are 0111, 1010, and 1110.

a. The first bits are 0, 1, and 1. The majority is 1.

b. The second bits are 1, 0, and 1. The majority is 1.

c. The third bits are 1, 1, and 1. The majority is 1.

d. The fourth bits are 1, 0, and 0. The majority is 0.

The result is 1110, or 0xE in hexadecimal.

SHA-512 Conditional Function calculation

- We apply the Conditional function on E, F, and G buffers. If the leftmost hexadecimal digits of these buffers are 0x9, 0xA, and 0xF respectively, what is the leftmost digit of the result?

Solution

$$(1 \text{ AND } 1) \oplus (\text{NOT } 1 \text{ AND } 1) = 1 \oplus 0 = 1$$

The digits in binary are 1001, 1010, and 1111.

a. The first bits are 1, 1,and 1.The result is F_1, which is 1.

b. The second bits are 0,0, and 1. The result is G_2, which is 1.

c. The third bits are 0,1,and 1.The result is G_3, which is 1.

d. The fourth bits are 1,0,and 1.The result is F_4, which is 0.

The result is 1110, or 0xE in hexadecimal.

Example using SHA-512

ASCII characters: “abc”, which is equivalent to the following 24-bit binary string:

01100001 01100010 01100011 = 616263 in Hexadecimal

The original length is 24 bits, or a hexadecimal value of 18.
the 1024-bit message block, in hexadecimal, is

6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018

$W0$ = 6162638000000000	$W5$ = 0000000000000000
$W1$ = 0000000000000000	$W6$ = 0000000000000000
$W2$ = 0000000000000000	$W7$ = 0000000000000000
$W3$ = 0000000000000000	$W8$ = 0000000000000000
$W4$ = 0000000000000000	$W9$ = 0000000000000000
$W10$ = 0000000000000000	$W13$ = 0000000000000000
$W11$ = 0000000000000000	$W14$ = 0000000000000000
$W12$ = 0000000000000000	$W15$ = 0000000000000018

Example using SHA-512

The following table shows the initial values of these variables and their values after each of the first two rounds.

a	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
c	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
d	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
g	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
h	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

The process continues through 80 rounds. The output of the final round is

73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9
d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

Example using SHA-512

The hash value is then calculated as

H1,7 =	5be0cd19137e2179	+ ceb9fc3691ce8326	= 2a9ac94fa54ca49f
H1,6 =	1f83d9abfb41bd6b	+ 25c96a7768fb2aa3	= 454d4423643ce80e
H1,5 =	9b05688c2b3e6c1f	+ 9bb4d39778c07f9e	= 36ba3c23a3feebbd
*H*1,4 =	510e527fade682d1	+ d08446aa79693ed7	= 2192992a274fc1a8
H1,3 =	a54ff53a5f1d36f1	+ 654ef9abec389ca9	= 0a9eeee64b55d39a
H1,2 =	3c6ef372fe94f82b	+ d67806db8b148677	= 12e6fa4e89a97ea2
H1,1 =	bb67ae8584caa73b	+ 10d9c4c4295599f6	= cc417349ae204131
H1,0 =	6a09e667f3bcc908	+ 73a54f399fa4b1b2	= ddaf35a193617aba

The resulting 512-bit message digest is

ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

SHA-512 Applications

- Used as part of a system to authenticate archival video from the International Criminal Tribunal of the Rwandan genocide.
- Proposed for use in DNSSEC
- are moving to 512-bit SHA-2 for secure password hashing by Unix and Linux vendors

SHA-512 Cryptanalysis

Published in	Year	Attack method	Rounds
New Collision Attacks Against Up To 24-step SHA-2	2008	Deterministic	24/80
Preimages for step-reduced SHA-2	2009	Meet-in-the-middle	42/80 46/80
Advanced meet-in-the-middle preimage attacks	2010	Meet-in-the-middle	42/80
Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family	2011	Biclique	50/80 57/80
Branching Heuristics in Differential Collision Search with Applications to SHA-512	2014	Heuristic differential	38/80

Summary

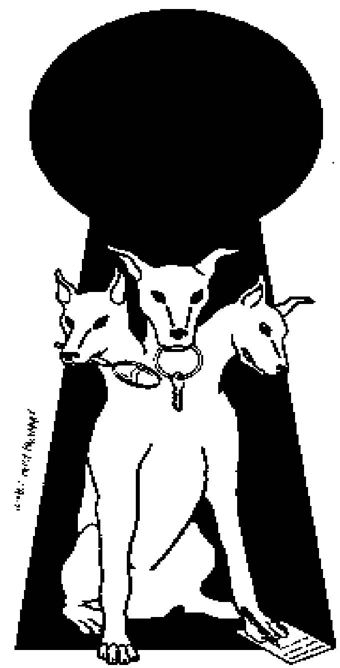
- Hash functions are keyless
  - Applications for digital signatures and in message authentication codes
- The three security requirements for hash functions are
  - one-wayness and collision resistance
- MD5 and SHA-0 is insecure
- Serious security weaknesses have been found in SHA-1
  - should be phased out
  - SHA-2 appears to be secure
  - May use SHA-512 and use the first 256 bytes
- SHA-3 (Secure Hash Algorithm 3) is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015

Authentication Applications

Outline

- Security Concerns
- Kerberos
- X.509 Authentication Service
- Recommended reading and Web Sites

KERBEROS



In Greek mythology, a many headed dog, the guardian of the entrance of Hades

KERBEROS

- Users wish to access services on servers.
- Three threats exist:
  - User pretend to be another user.
  - User alter the network address of a workstation.
  - User eavesdrop on exchanges and use a replay attack.

KERBEROS

- Provides a centralized authentication server to authenticate users to servers and servers to users.
- Relies on conventional encryption, making no use of public-key encryption
- Two versions: version 4 and 5
- Version 4 makes use of DES

Kerberos Version 4

- Terms:
  - C = Client
  - AS = authentication server
  - V = server
  - ID_c = identifier of user on C
  - ID_v = identifier of V
  - P_c = password of user on C
  - AD_c = network address of C
  - K_v = secret encryption key shared by AS an V
  - TS = timestamp
  - || = concatenation

A Simple Authentication Dialogue

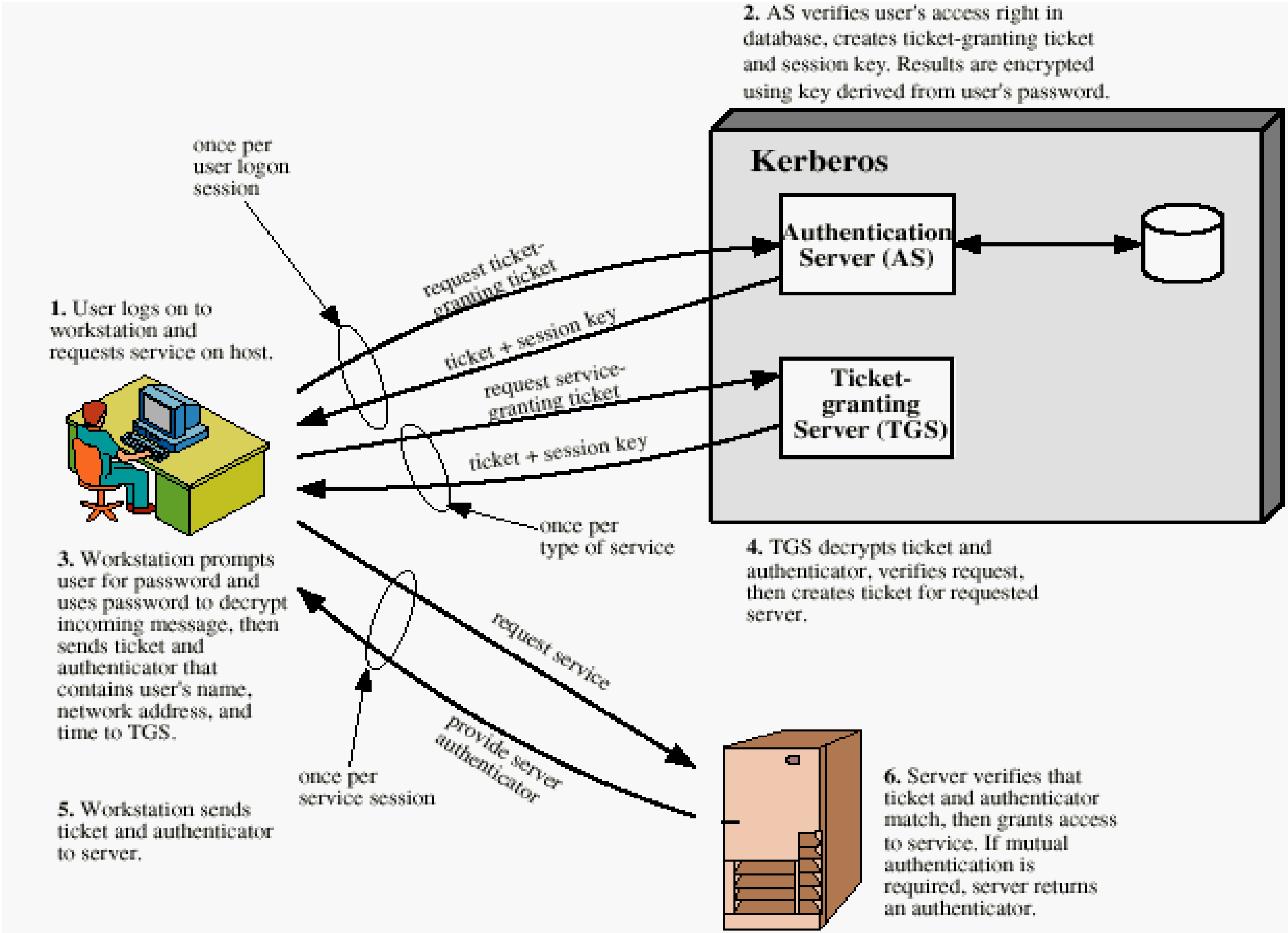
(1) $C \rightarrow AS$: $ID_c \,||\, P_c \,||\, ID_v$

(2) $AS \rightarrow C$: Ticket

(3) $C \rightarrow V$: $ID_c \,||$ Ticket

Ticket = $E_{K_v}[ID_c \,||\, P_c \,||\, ID_v]$

Overview of Kerberos



Request for Service in Another Realm

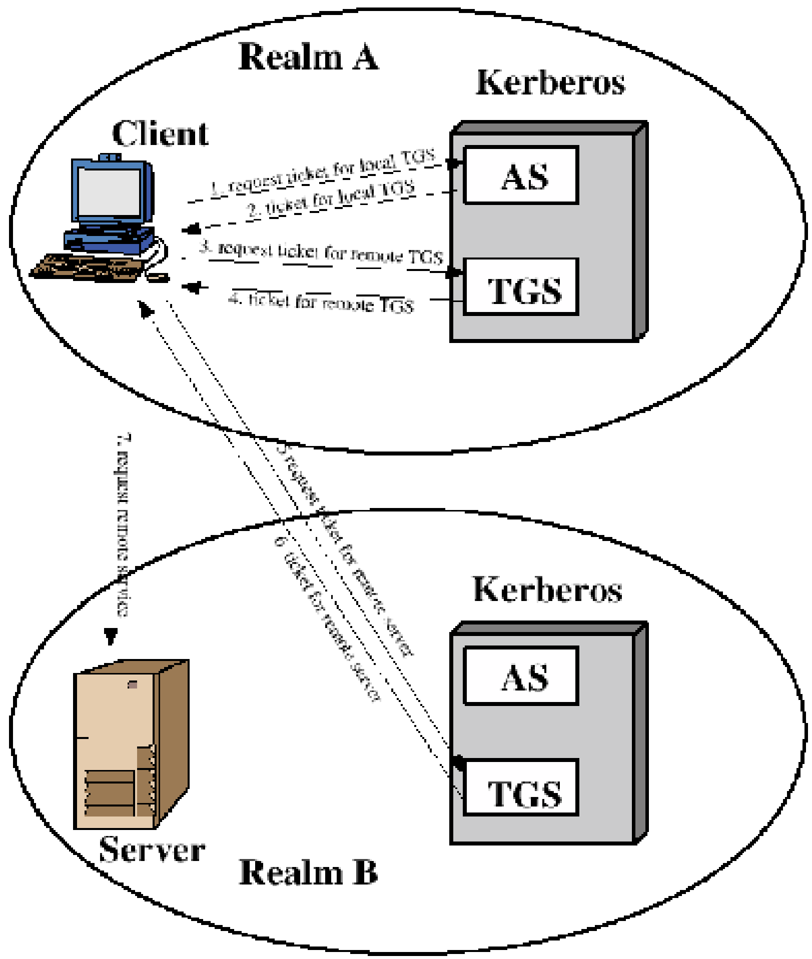


Figure 4.2 Request for Service in Another Realm

Difference Between Version 4 and 5

- Encryption system dependence (V.4 DES)
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Interrealm authentication

Kerberos Encryption Techniques

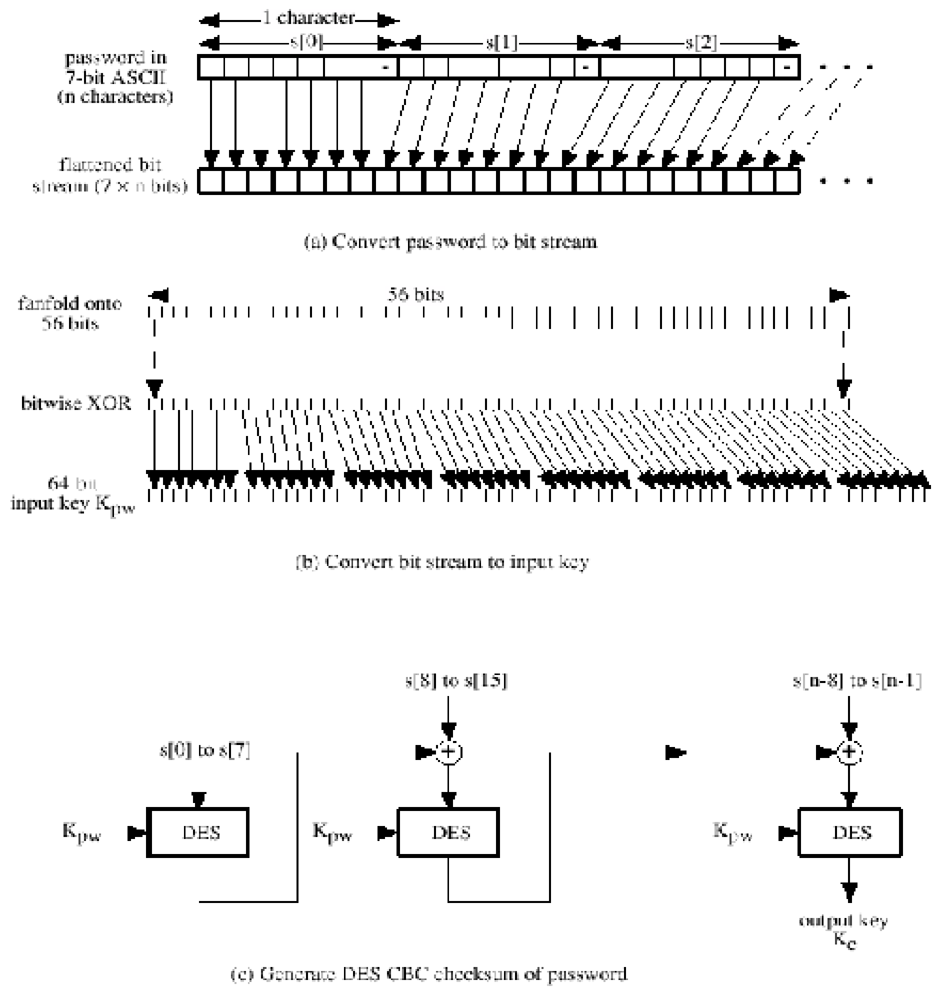


Figure 4.6 Generation of Encryption Key from Password

PCBC Mode

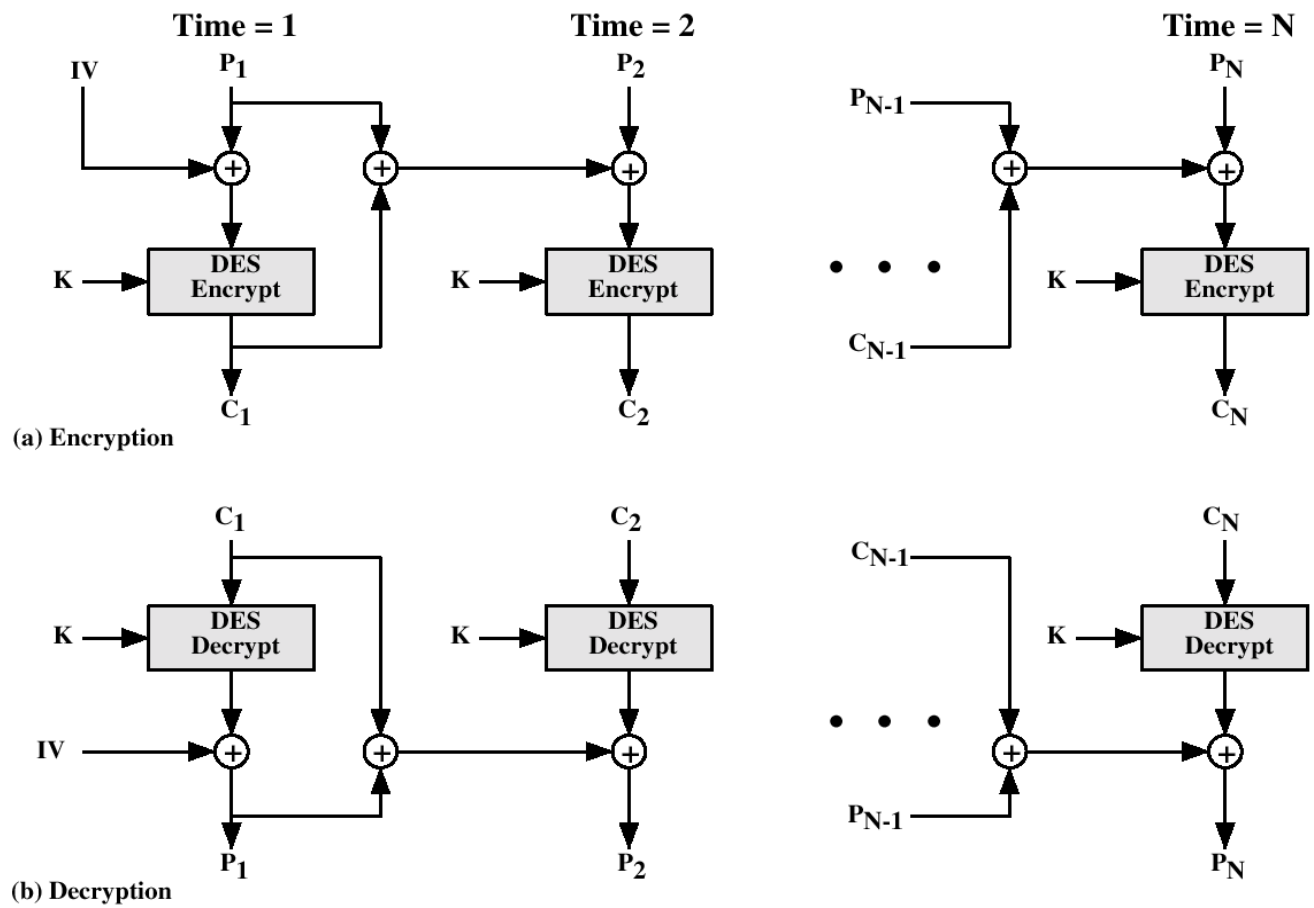


Figure 4.7 Propagating Cipher Block Chaining (PCBC) Mode

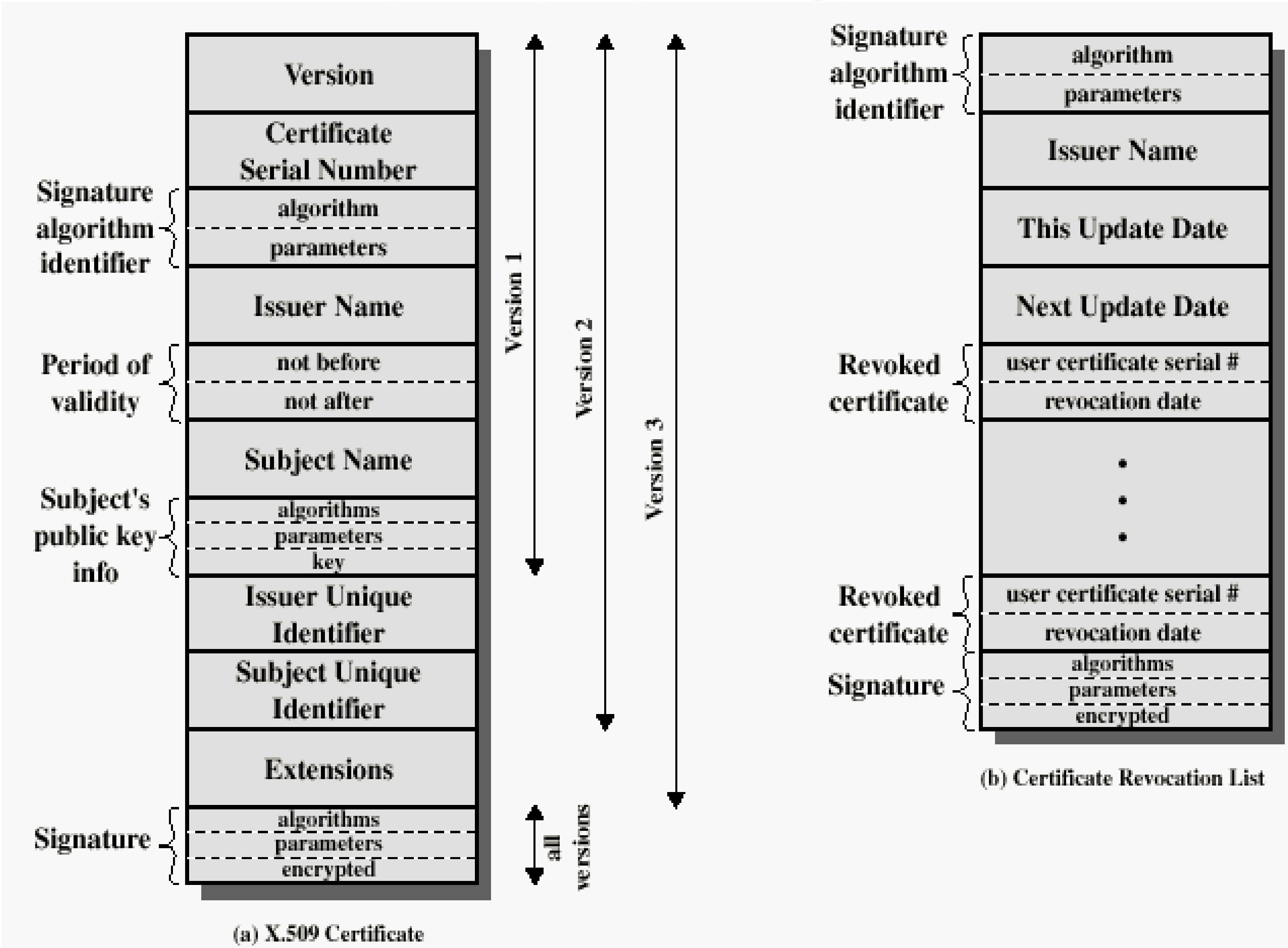
Kerberos - in practice

- **Currently have two Kerberos versions:**
- 4 : restricted to a single realm
- 5 : allows inter-realm authentication, in beta test
- Kerberos v5 is an Internet standard
- specified in RFC1510, and used by many utilities
- **To use Kerberos:**
- need to have a KDC on your network
- need to have Kerberised applications running on all participating systems
- major problem - US export restrictions
- Kerberos cannot be directly distributed outside the US in source format (& binary versions must obscure crypto routine entry points and have no encryption)
- else crypto libraries must be reimplemented locally

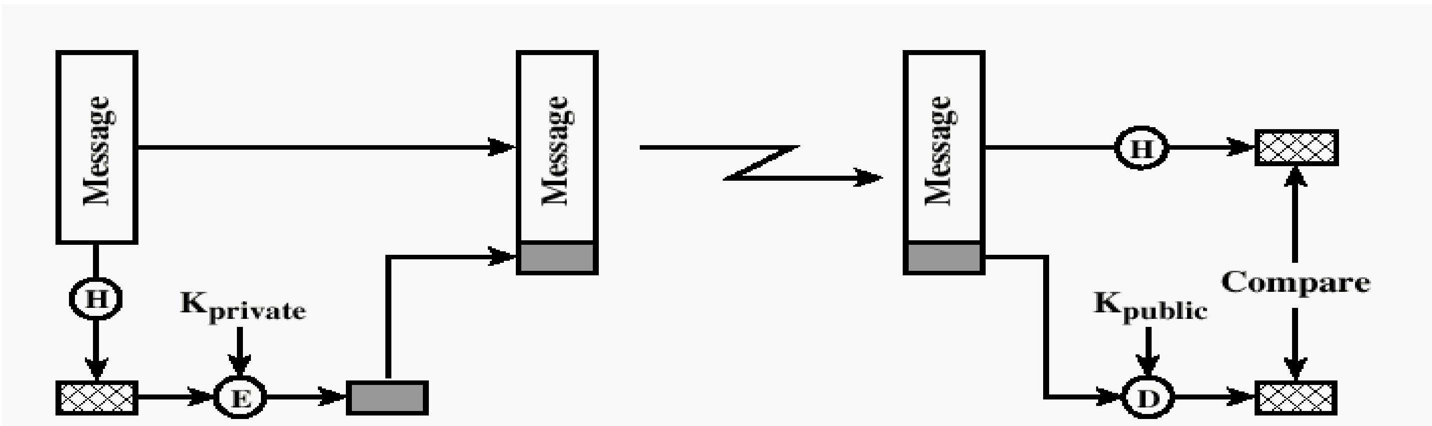
X.509 Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use.

X.509 Formats



Typical Digital Signature Approach

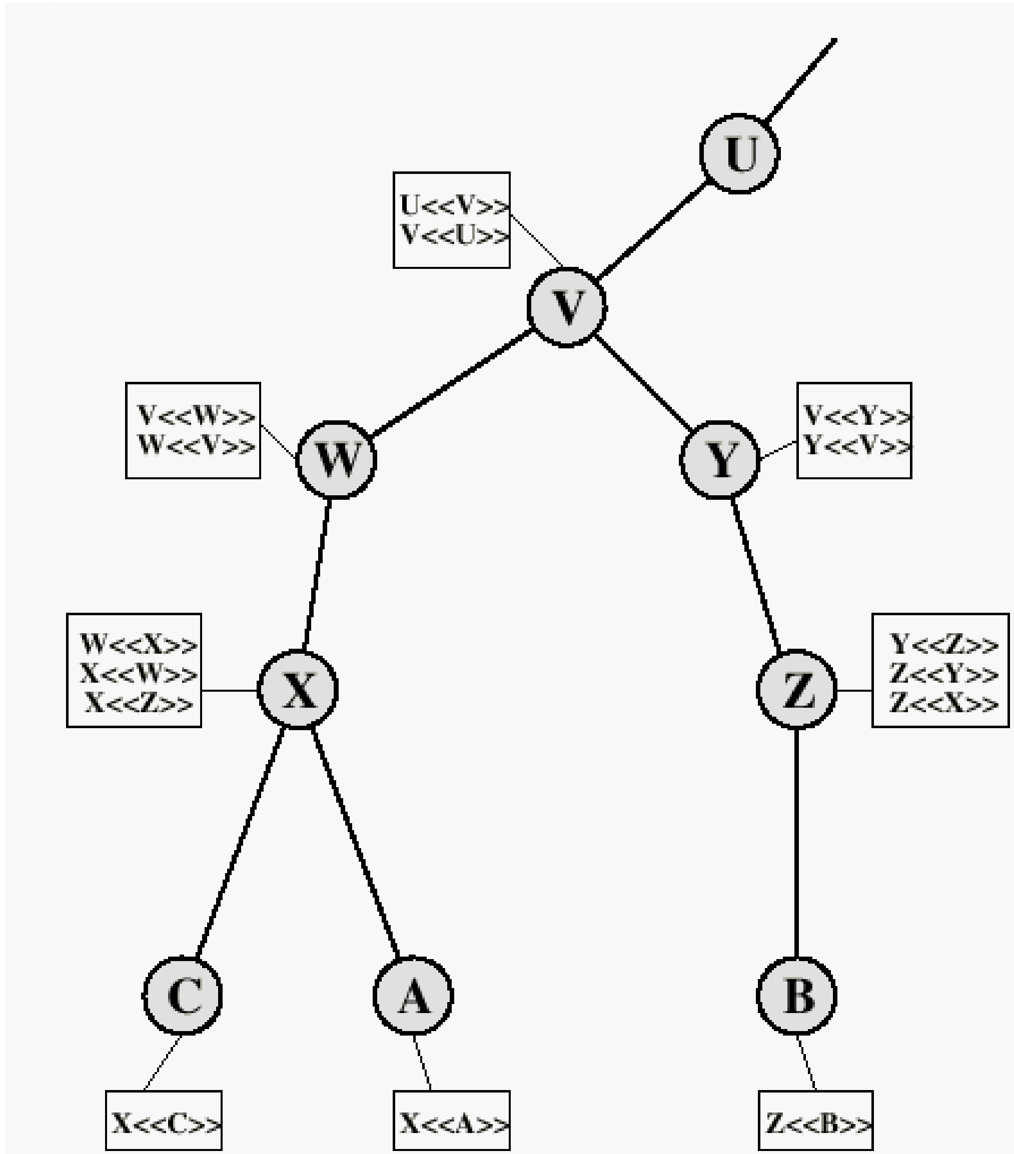


(b) Using public-key encryption

Obtaining a User's Certificate

- Characteristics of certificates generated by CA:
  - Any user with access to the public key of the CA can recover the user public key that was certified.
  - No part other than the CA can modify the certificate without this being detected.

X.509 CA Hierarchy



Revocation of Certificates

- Reasons for revocation:
  - The users secret key is assumed to be compromised.
  - The user is no longer certified by this CA.
  - The CA's certificate is assumed to be compromised.

Authentication Procedures

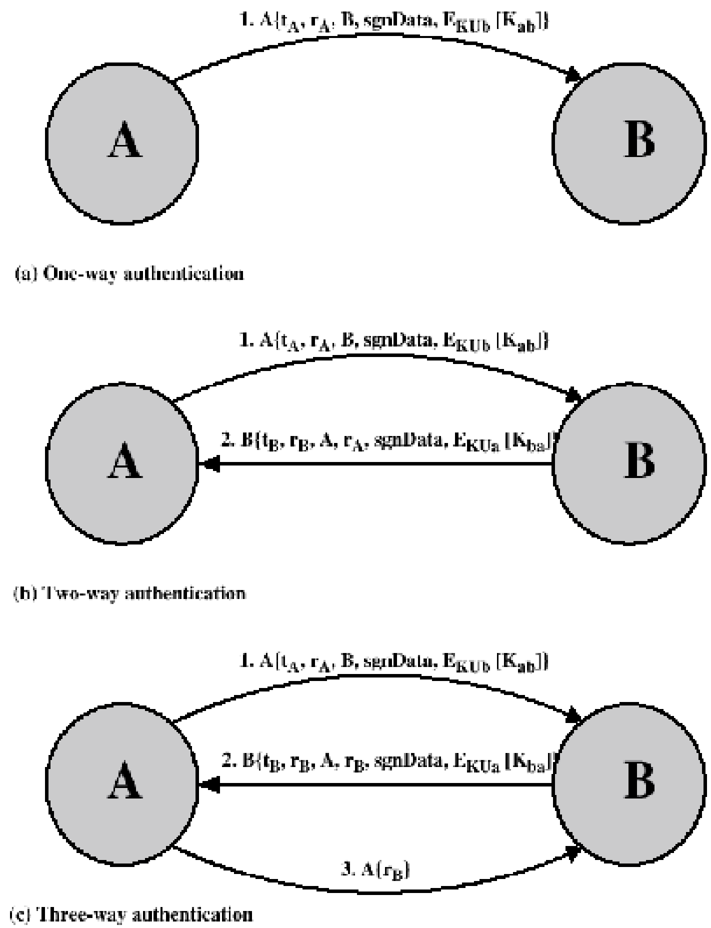


Figure 4.5 X.509 Strong Authentication Procedures

Recommended Reading and WEB Sites

- www.whatis.com (search for kerberos)
- Bryant, W. Designing an Authentication System: A Dialogue in Four Scenes. http://web.mit.edu/kerberos/www/dialogue.html
- Kohl, J.; Neuman, B. "The Evolotion of the Kerberos Authentication Service" http://web.mit.edu/kerberos/www/papers.html
- http://www.isi.edu/gost/info/kerberos/