Unit-3

Cryptographic Hash Functions,

Message Authentication Codes

&

Key Management and Distribution

By Dr Ravindra Babu Kallam

- **Cryptographic Hash Functions:**
  - Message Authentication
  - Secure Hash Algorithm (SHA-512)
- **Message authentication codes:**
  - Authentication requirements,
  - HMAC,
  - CMAC,
  - Digital signatures,
  - Elgamal Digital Signature Scheme.
- **Key Management and Distribution:**
  - Symmetric Key Distribution Using Symmetric & Asymmetric Encryption,
  - Distribution of Public Keys,
  - Kerberos, X.509 Authentication Service,
  - Public – Key Infrastructure

Approaches to message authentication:

- A message ,file, document or other collection of data is said to be authentic when it is genuine and came from its original source.

- Message authentication is a procedure that allows communicating parties to verify that the received messages are authentic.

Message authentication requirements

In the context of communications across a network, the following attacks can be identified.

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined.

   In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent source.
4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6.**Timing modification:** Delay or replay of messages.

7. **Source repudiation:** Denial of transmission of message by source.

8. **Destination repudiation:** Denial of receipt of message by destination.

- ➢ Measures to deal with the first two attacks are in the realm of message confidentiality.
- ➢ Measures to deal with items 3 through 6 are generally regarded as message authentication.
- ➢ Mechanisms for dealing with item 7 come under the heading of digital signatures.
- ➢ Dealing with item 8 may require a combination of the use of digital signatures and a protocol designed to counter this attack.

- Authentication using conventional Encryption:
  - It is possible to perform authentication simply by using conventional encryption, if we assume that the key is secured between sender and receiver.
- Message Authentication without Message Encryption:
  - There are several approaches to message authentication that do not rely on encryption.
  - It is specially used where only authentication is required but not the confidentiality, few of such situations are:
    - **When same message is broadcasted to number of destinations**
    - **An exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. *Authentication is carried out on a selective basis, messages are chosen at random for checking.***
    - **Authentication of a computer program in plain text is an attractive service. The computer program can be executed with out having to decrypt it every time.**

Message Authentication Code:

- Authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code, that is appended to the message.
- This technique assumes that two communicating parties, say A and B, share a common secrete key K_{AB}.
- When A has a message to send to B, it calculates the message authentication code as a function of the message and the key: $MAC_M = F(K_{AB}, M)$.
- The message plus code are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.
- The received MAC is compared with the new MAC as shown in the fig. if both are same then the message is authenticated.

Message authentication using MAC

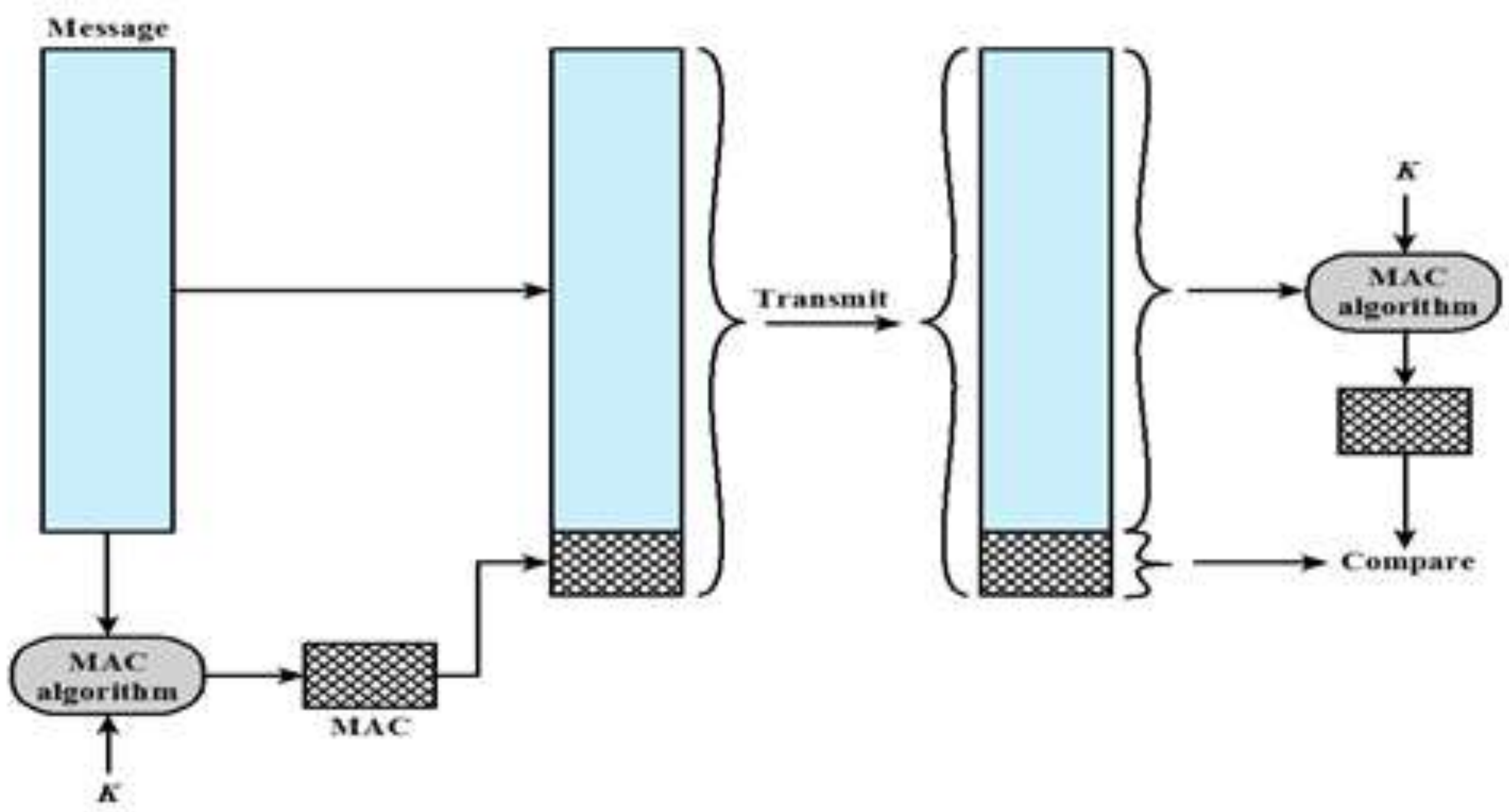
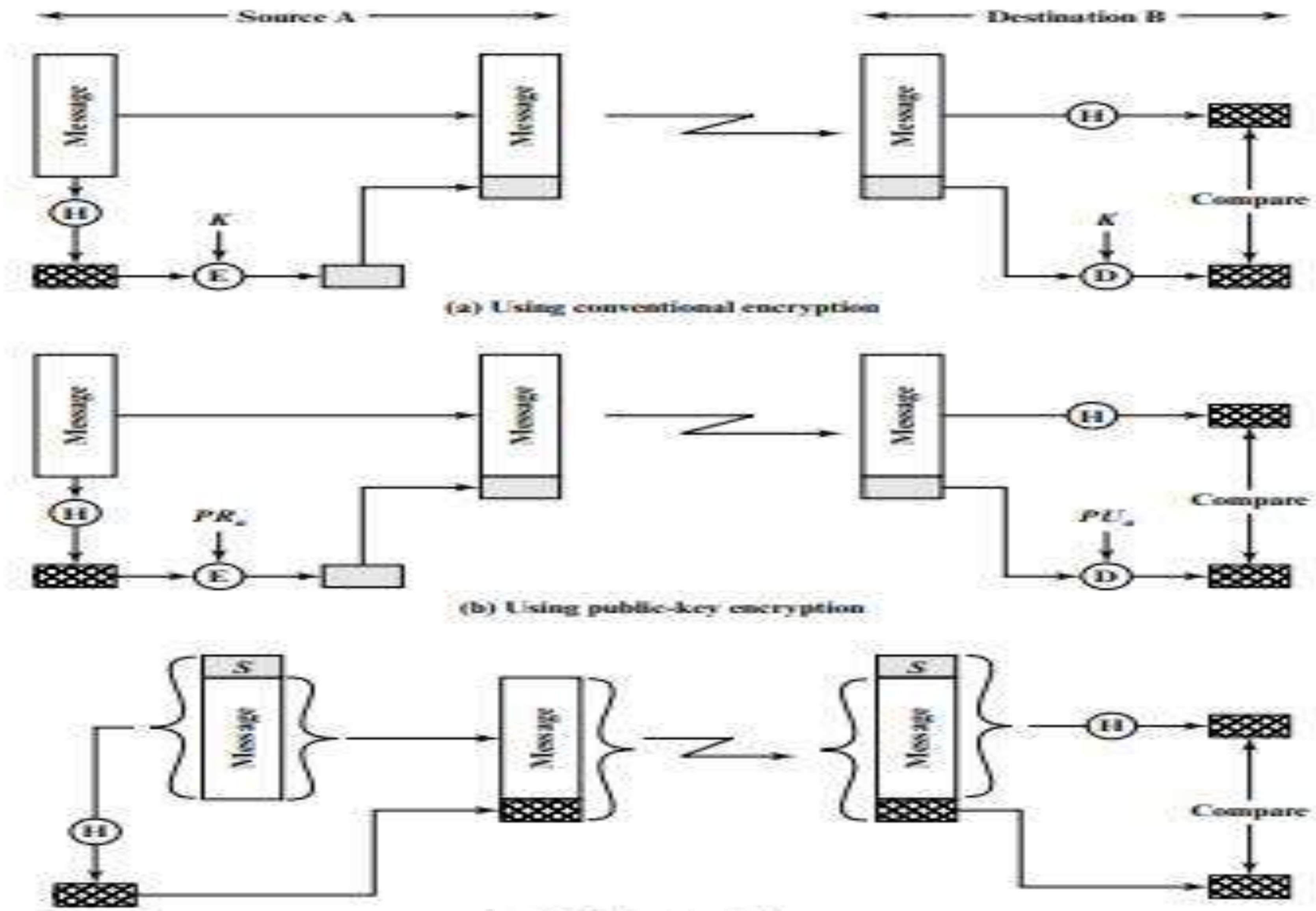


Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).

One way Hash Function:

- A hash function accepts a variable size message M as input and produces a fixed- size message digest H(M) as output. Unlike MAC ,a hash function does not take the secret key as input.
- To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.
- As shown in the fig there are three ways in which the message can be authenticated.
  - Using Conventional encryption
  - Using public key encryption
  - Using secrete value. Several reasons for using this technique are:
    - Encryption software is quit slow.
    - Encryption hardware cost are non negligible
    - Encryption hardware is optimized toward large data size
    - Encryption algorithms may be covered by patents
    - Encryption algorithms may be subject to export control.

Message authentication using One-way Hash Function



Source A
Destination B
Message
H
K
E
D
Compare
(a) Using conventional encryption
PR_a
PU_a
(b) Using public-key encryption
S

Capt RB Kallam

Figure 3.2 Message Authentication Using a One-Way Hash Function

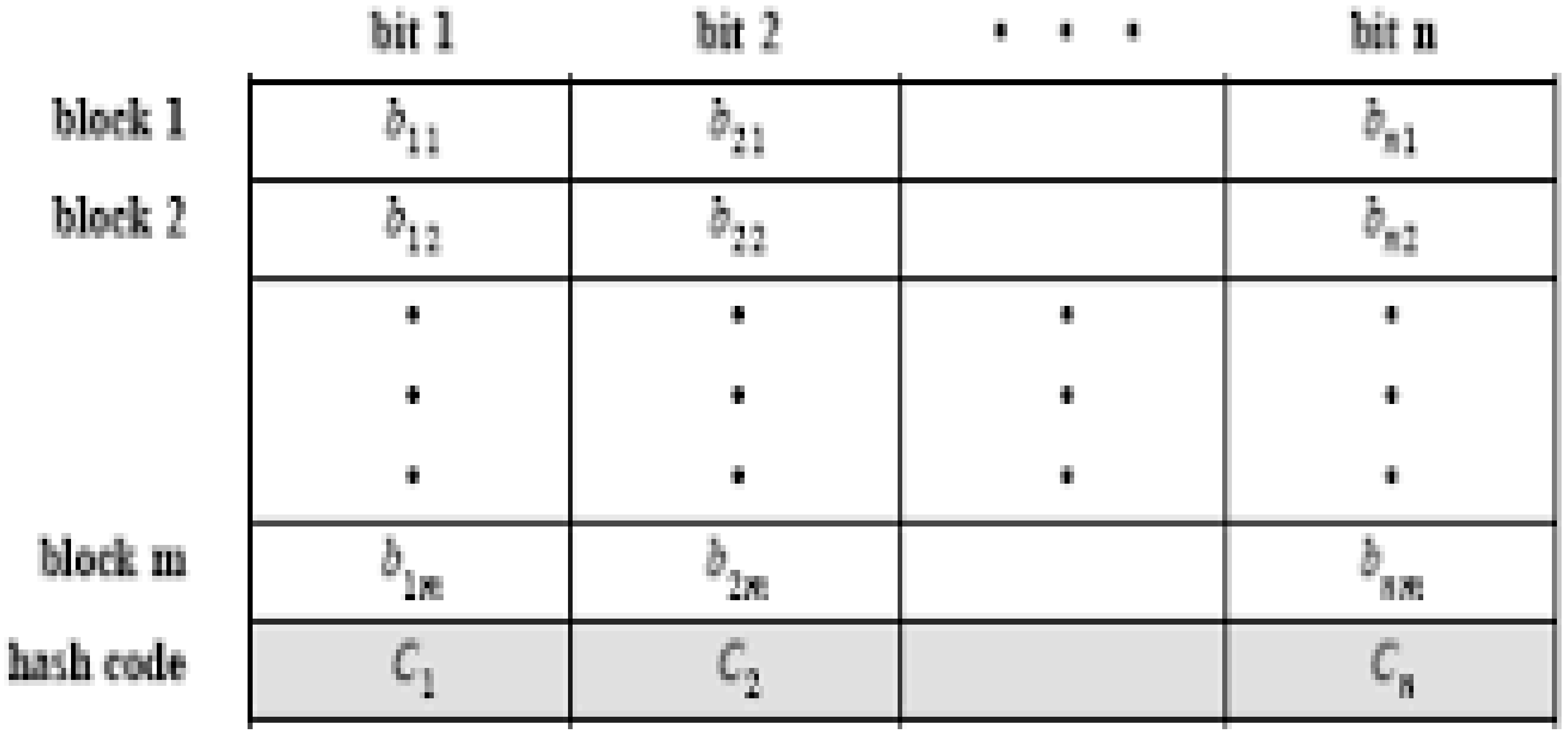
Secure Hash Function and HMAC

- One way hash function or secure hash function, is important not only in message authentication but in digital signatures.
- Hash Function Requirements:
  - H can be applied to a block of data of any size
  - H produces a fixed length output
  - $H(x)$ is relatively easy to compute for any given x, making both hardware and software implementation practical.
  - For any given code h, it is computationally infeasible to find x such that $H(x) = h$
  - For any given block x, it is computationally infeasible to find $y=x$ with $H(y) = H(x)$.

Simple Hash Functions:

- Simplest hash function is the bit-by-bit exclusive – OR (XOR) of every block.
- This can be expressed as follows:
  - $C_i = b_{i1} \oplus b_{i2} \oplus - - - - \oplus b_{im}$
  - Where
    - C_i = ith bit of the hash code, 1<=i<=n
    - m = number of n-bit blocks in the input
    - b_{ij} = ith bit in jth block
    - $\oplus$ = XOR operation

Simple Hash Function Using Bitwise XOR



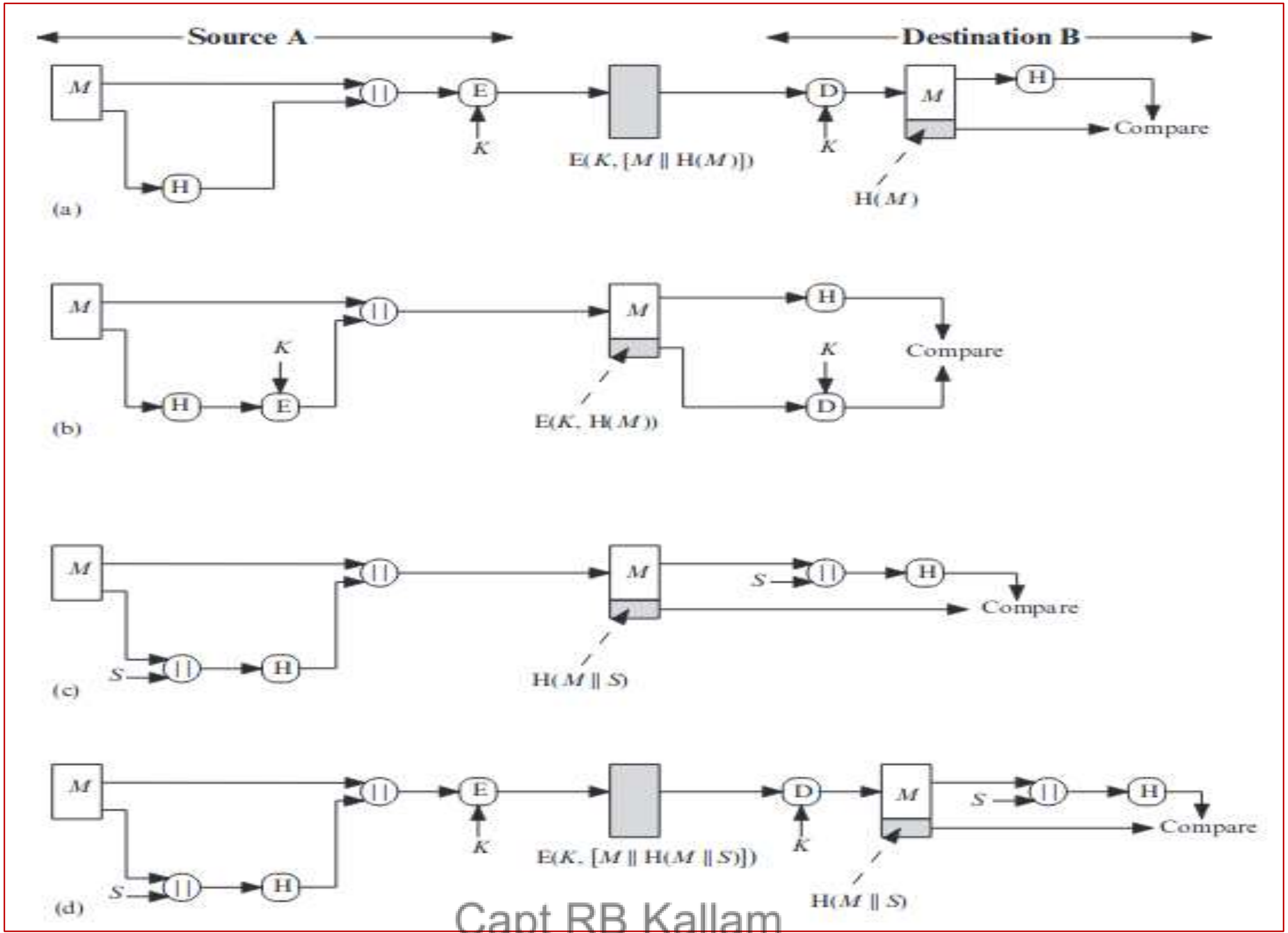
	bit 1	bit 2	• • •	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

Figure 3.3 Simple Hash Function Using Bitwise XOR

Applications of cryptographic Has Functions

- ❖ Message Authentication: Message authentication is a mechanism or service used to verify the integrity of a message.
- ❖ Digital Signature: The operation of the digital signature is similar to that of the MAC. In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

Use of Hash function for Message Authentication



a) Message plus concatenated hash code is encrypted using symmetric encryption.
b) Only the hash code is encrypted, using symmetric encryption
c) Use a hash function with some secret value (s) but no encryption for massage authentication
d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code

Use of hash function for Digital signatures

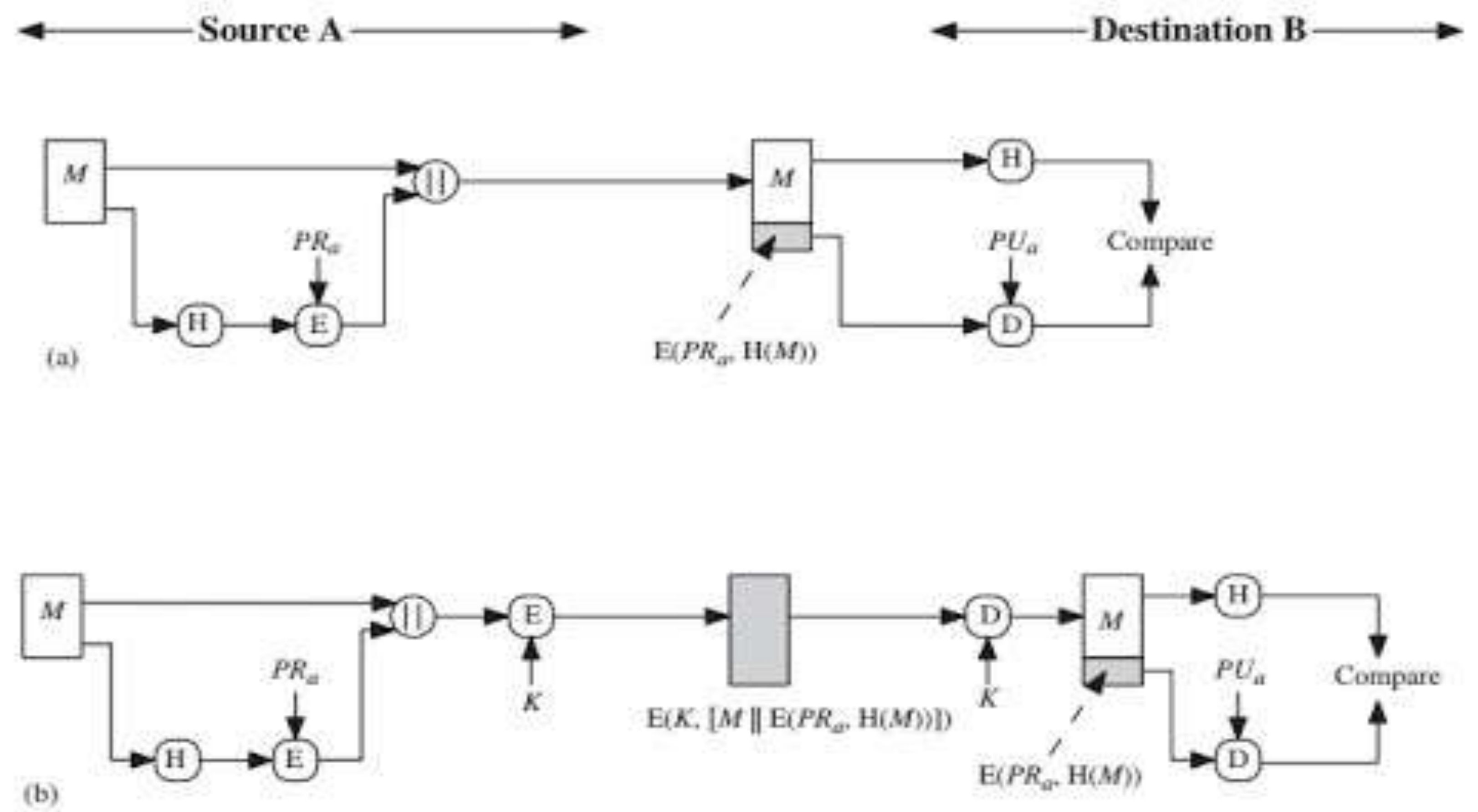


Figure 11.3 Simplified Examples of Digital Signatures

a) Using public key cryptography, Encrypt the hash code with senders private key to achieve digital signature.

b) If the confidentiality as well as digital signature is desired, then the message plus the private key encrypted hash code can be encrypted using a symmetric key as shown in the figure.

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- based on design of MD4 with key differences
- produces 160-bit hash values

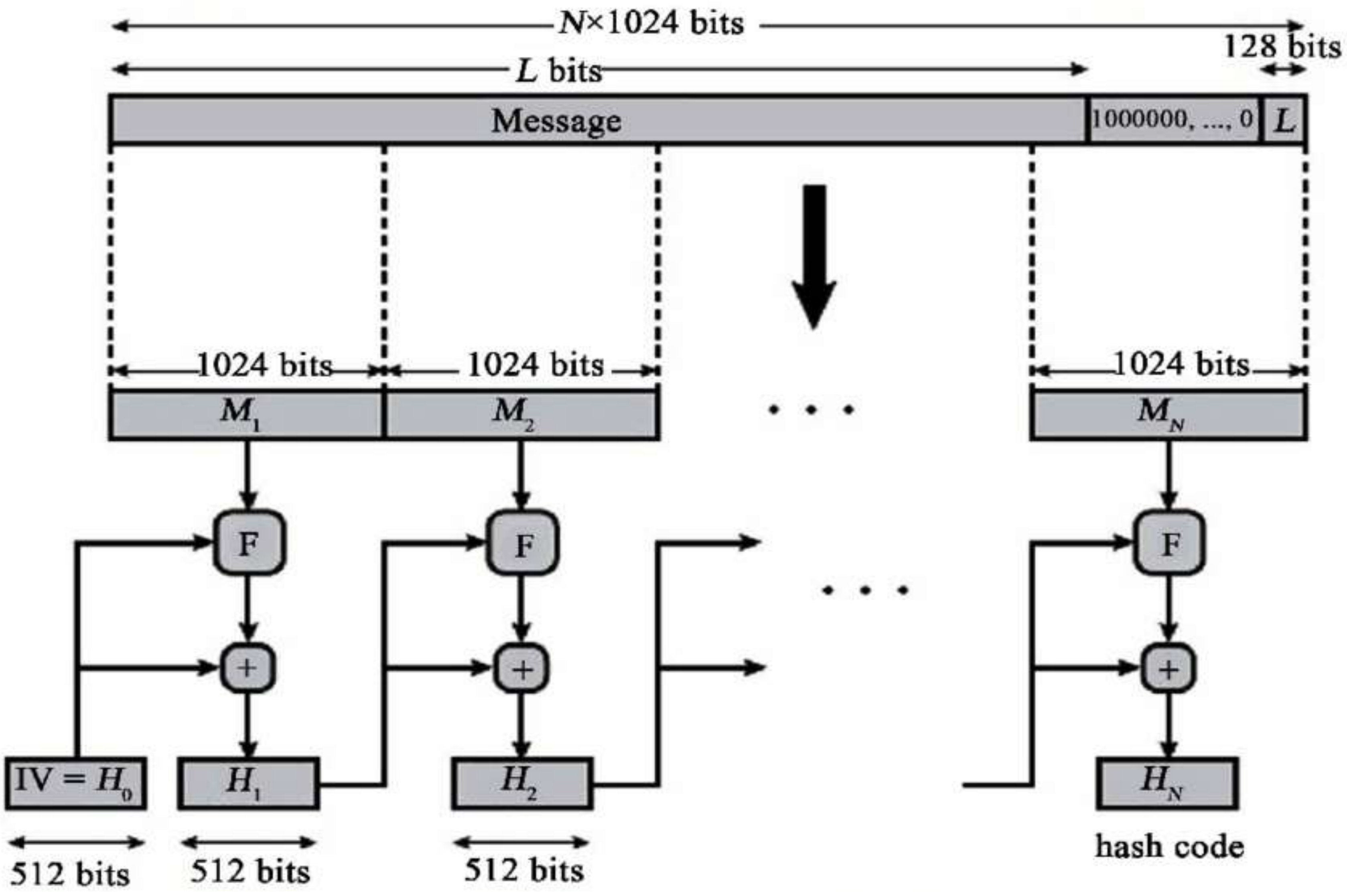
Revised Secure Hash Standard:

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

Comparison of SHA parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps/ rounds	80	64	80	80

Message Digest Generation Using SHA-512



Capt RB Kallam

Step1: Append padding bits. The message is padded so that its length in bits is 896 modulo 1024. That is, the length of the padding message is 128 bits less then an integer multiple of 1024 bits.

The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step2: Append length. A 128 bit representation of the length in bits of the original message is appended to the result of step1.

The out come of the first two steps yields a message that is an integer multiple of 1024 bits in length.

The expanded message is represented as a sequence of 1024-bit blocks M_1 , $M_{2,}$ ….. $M_{N,}$ so that the total length of the expanded message is N x 1024 bits.

Step3: Initialize MD buffer.

A 512 bit buffer is used to hold intermediate and final results of the hash function.

The buffer can be represented as a **eight 64-bit registers** (a,b,c,d,e,f,g,h).

The registers are initialized to the following 64-bit integers (Hex Values)

a = 6A09E667F3BCC908	e = 510E527FADE682D1
b = BB67AE8584CAA73B	f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CDI9137E2179

These values are stored in *Big - endian format*, *which is the most significant byte of a word in the low address byte position*.

The initialization values appear as follows:

a = 6A09E667F3BCC908	e = 510E527FADE682D1
b = BB67AE8584CAA73B	f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CDI9137E2179

Note: these words are obtained by taking the first 64 bits of the fractional parts of the square roots of the first 8 prime numbers.

Step4: Process message in 1024-bit (128 word) blocks.

The heart of the algorithm is a module that consists of 80 rounds.

Each round takes as input the 512 bit buffer value a,b,c,d,e,f,g and h, and updates the contents of the buffer.

At input to the first round, the buffer has a value of the intermediate hash value, $H_{i-1.}$ Each round t makes use of a 64 bit value W_t, derived from the current 1024 bit block being processed (M_i).

Each round also makes use of an additive constant Kt, where 0<=t<=79 indicates one of the 80 rounds.

The output of the 80th round is added to the input to the first round (H_{i-1}) to produce (H_i) .

The addition is done independently for each of the eight words in the buffer with each of the corresponding words in H_{i-1}.

Step5: Output.

- After all N 1024-bit blocks have been processed, the output from the Nth the stage is the 512-bit message digest.

.

SHA-512 Processing of a Single 1024 bit Block

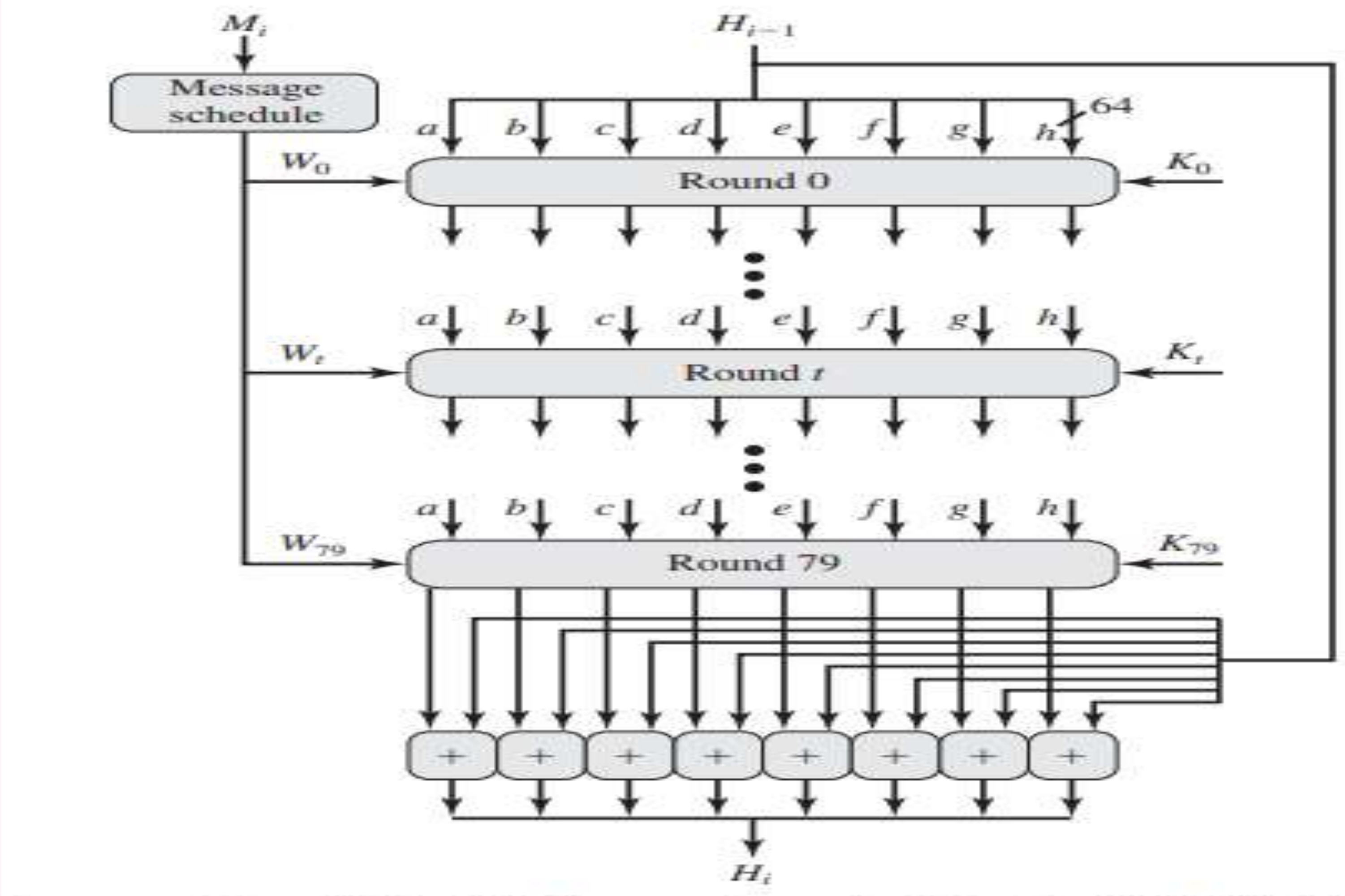


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

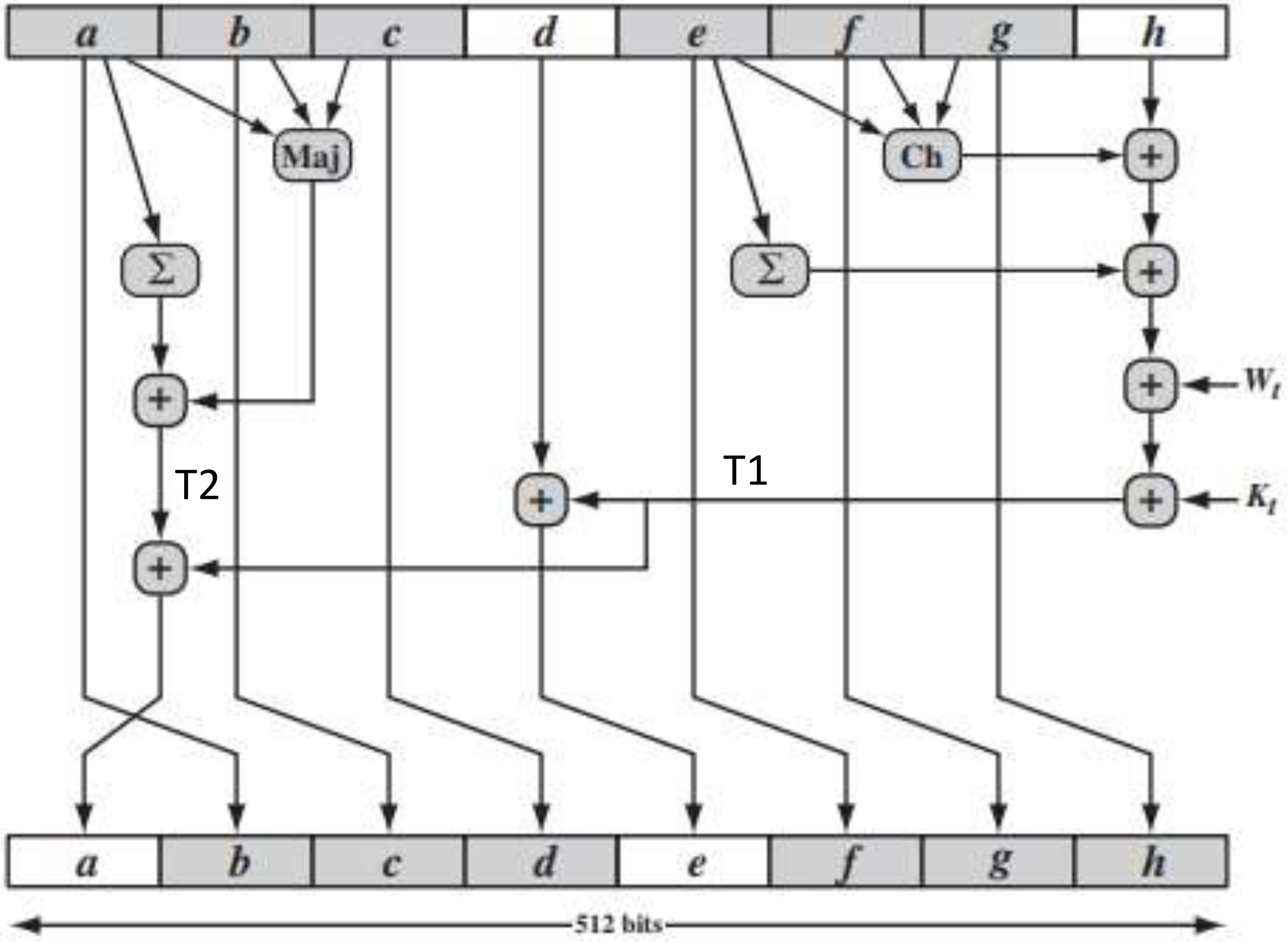


Figure 11.11 Elementary SHA-512 Operation (single round)

Let us look in more detail at the logic in each of the 80 steps of the processing of one 512-bit block (Figure 11.11). Each round is defined by the following set of equations:

$$
\begin{aligned}
T_1 &= h + \mathrm{Ch}(e, f, g) + \left(\textstyle\sum_1^{512} e\right) + W_t + K_t \\
T_2 &= \left(\textstyle\sum_0^{512} a\right) + \mathrm{Maj}(a, b, c) \\
h &= g \\
g &= f \\
f &= e \\
e &= d + T_1 \\
d &= c \\
c &= b \\
b &= a \\
a &= T_1 + T_2
\end{aligned}
$$

where :
t = step number; 0 … t … 79
Ch(e, f, g) = (e AND f) $\oplus$ (NOT e AND g) the conditional function:
 If e then f else g
Maj(a, b, c) = (a AND b) $\oplus$ (a AND c) $\oplus$ (b AND c)
 the function is true only of the majority (two or three) of the arguments are true

$$\left(\Sigma_0^{512} a\right) = \mathrm{ROTR}^{28}(a) \oplus \mathrm{ROTR}^{34}(a) \oplus \mathrm{ROTR}^{39}(a)$$
$$\left(\Sigma_1^{512} e\right) = \mathrm{ROTR}^{14}(e) \oplus \mathrm{ROTR}^{18}(e) \oplus \mathrm{ROTR}^{41}(e)$$
$$\mathrm{ROTR}^{n}(x) = \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits}$$

W t = a 64-bit word derived from the current 1024-bit input block
Kt = a 64-bit additive constant
+ = addition modulo 2^{64} .
Two observations can be made about the round function.

1. Six of the eight words of the output of the round function involve simply permutation (b, c, d, f, g, h) by means of rotation. This is indicated by shading in Figure 11.11.
2. Only two of the output words (a, e) are generated by substitution.

Creation of 80- word input sequence for SHA 512 processing of single block

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

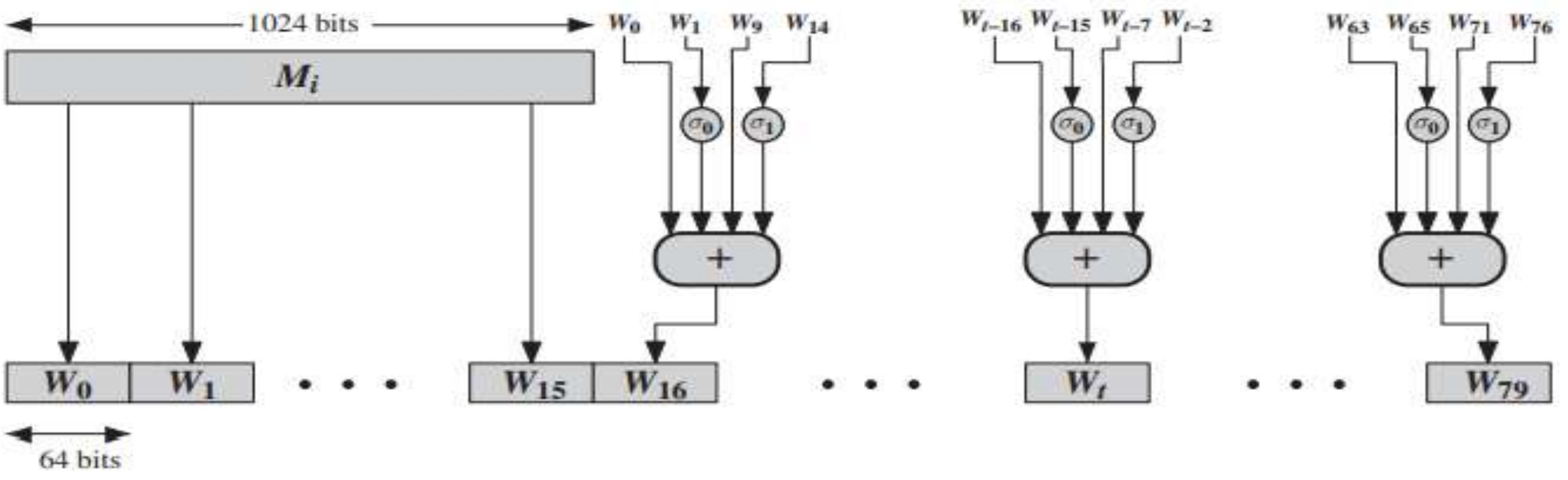
$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$

$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

$\text{SHR}^n(x)$ = left shift of the 64-bit argument x by n bits with padding by zeros on the right

$+$ = addition modulo 2^{64}



1024 bits
M_i
W_0 W_1 W_9 W_{14}
W_{t-16} W_{t-15} W_{t-7} W_{t-2}
W_{63} W_{65} W_{71} W_{76}
σ_0
σ_1
+
W_0 W_1 W_{15} W_{16} W_t W_{79}
64 bits

Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

Capt RB Kallam

HMAC:

HMAC design objectives:

- To use with out modifications, available hash functions such as MD5 and SHA -1.
- In particular, to use hash functions that perform well in software and for which code is freely and widely available.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function with out incurring a significant degradation.
- To use and handle keys in a simple way.
- Easier for cryptanalysis.

HMAC Algorithm:

- Fig., below shows the overall operation of HMAC.
- In this:

H = embedded hash function (e.g., SHA -1)

IV = initial value input to hash function

M = message input to HMAC

Yi = i th block of M, 0<= i <= (L-1)

L = number of blocks in M

b = number of bits in a block

n = length of hash code produced by embedded hash function

K = secrete key recommended length is >=n;

K+ = K padded with zeros on the left so that the result is b- bits in length

ipad = 00110110 (36 in hex) repeated b/8 times.

opad= 01011100 (5C in hex) repeated b/8 times.

Then HMAC can be expressed as :

HMAC (K,M) = H[(K^+ XOR opad) II H[(K^+ XOR ipad) II M]]

- Steps in words:
  1. Append zeros to the left end of K to create a b bit string K^+
  2. XOR K^+ with ipad to produce the b –bit block Si
  3. Append M to Si
  4. Apply H to the stream generated in step3
  5. XOR K^+ with opad to produce the b –bit block S_0
  6. Append the hash result from step 4 to S_0
  7. Apply H to the stream generated in step 6 and output the result.

HMAC Overview

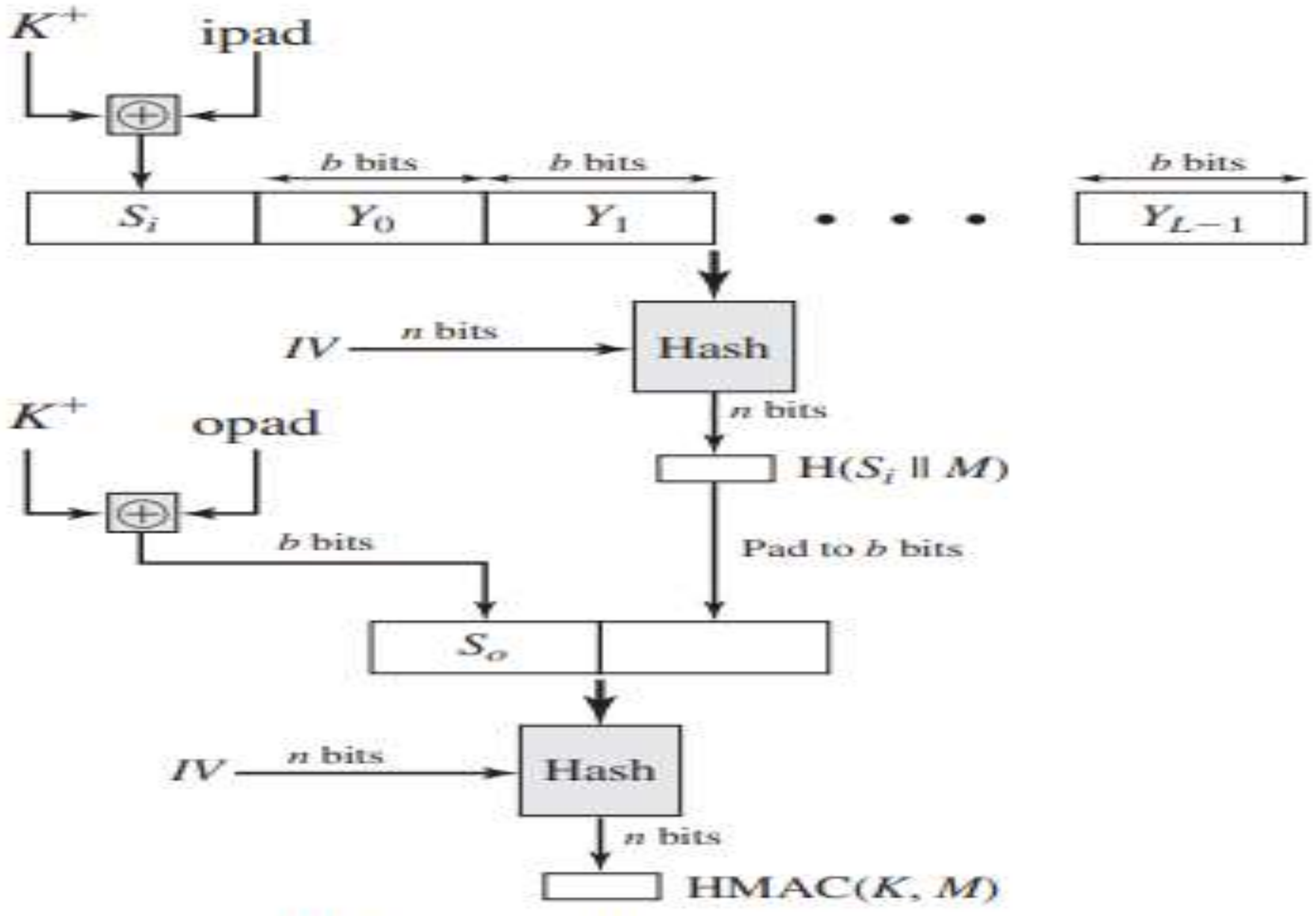


Figure 12.5 **HMAC Structure**

Efficient Implementation of HMAC

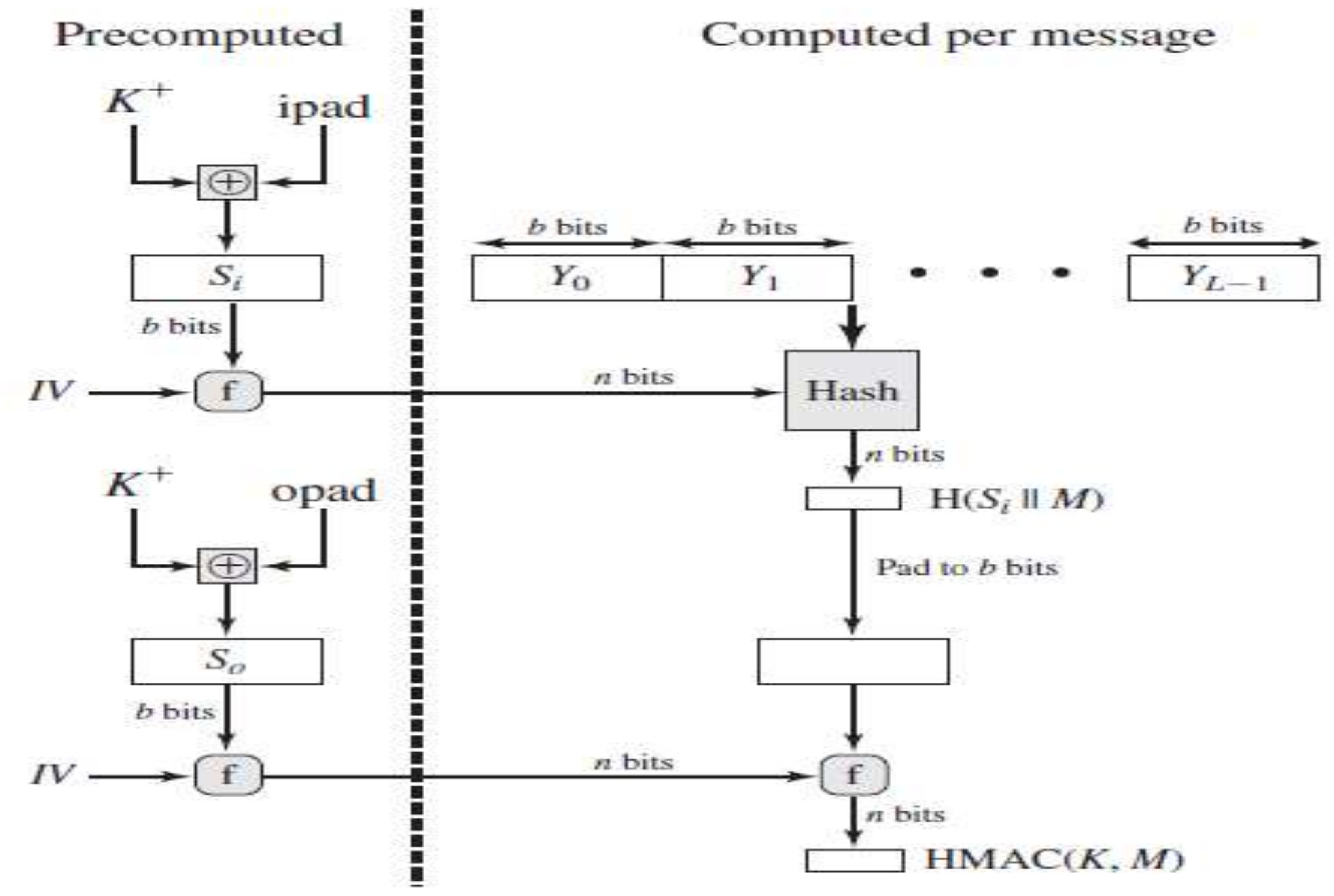


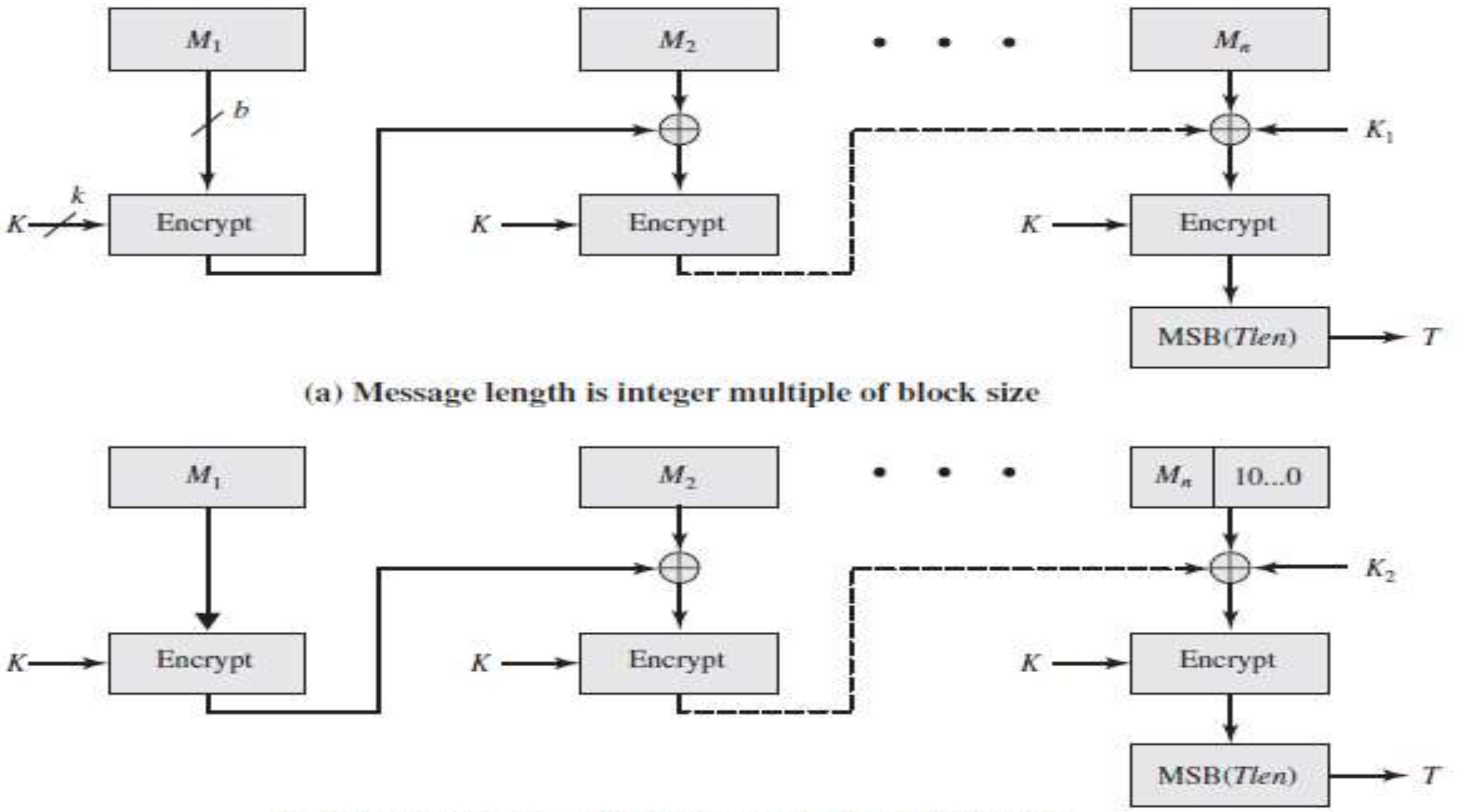
Figure 12.6 Efficient Implementation of HMAC

Cipher based MAC (CMAC)

- CMAC is widely used in government and industry.
- It has the message size limitation
- Cipher based MACs are block cipher based message authentication codes.
- In CMAC, the message size is an integer multiple ‘n’ of cipher block length ‘b’. For AES, b=128 and for triple DES, b=64. The msg is divided into ‘n’ blocks, M1,M2,….Mn.
- It uses ‘k’ bit encryption key K and an n-bit constant K1.
- CBC encrypt the Msg and retain the result of the last block encryption as the computed MAC value.
- If the msg is not in integer multiple of the cipher block, then the final block is padded to the right with a 1 and as many 0s as necessary., so that the final block is also of length ‘b’.

- $C_1= E(K,M_1)$
- $C_2= E(K,[M_2 XOR C_1)$
- $C_3= E(K,[M_3 XOR C_2)$
-
-
- $C_n= E(K,[M_N XOR C_{n-1} XOR K_1)$
- $T= MSB_{Tlen}(C_n)$
- Where T=message authentication code, also referred to as the tag.
- *Tlen= bit length of T*
- *$MSB_S(X)$= the s leftmost bits of the bit string X.*

Cipher based MAC (CMAC)



(a) Message length is integer multiple of block size

(b) Message length is not integer multiple of block size

Figure 12.8 Cipher-Based Message Authentication Code (CMAC)

Capt RB Kallam

Digital Signature:

- A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature.
- This is possible by encrypting the message with the creator's private key.

  The signature guaranties the source and integrity of the message.

Digital Signature:

Disputes in simple message authentication:

- B can forge a different message and clime that it came from A
- A can deny sending the message. Because it is possible for B to forge A’s message, there is no way to prove that A have not sent it.
- hence include authentication function with additional capabilities

Properties:

- verify author, date & time of signature
- It must to authenticate the contents at the time of the signature
- It must be verifiable by third parties, to resolve disputes

Digital Signature:

Requirements:

- Signature must depend on the message being signed
- Signature must use some information unique to sender
  - to prevent both forgery and denial
- It must be relatively easy to produce
- It must be relatively easy to recognize & verify
- It must be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- It must be practical to retain a copy of the digital signature in storage

Digital Signature:

Digital signatures fall in two categories:

- Direct digital signatures
- Arbitrated digital signatures.

Direct Digital Signatures:

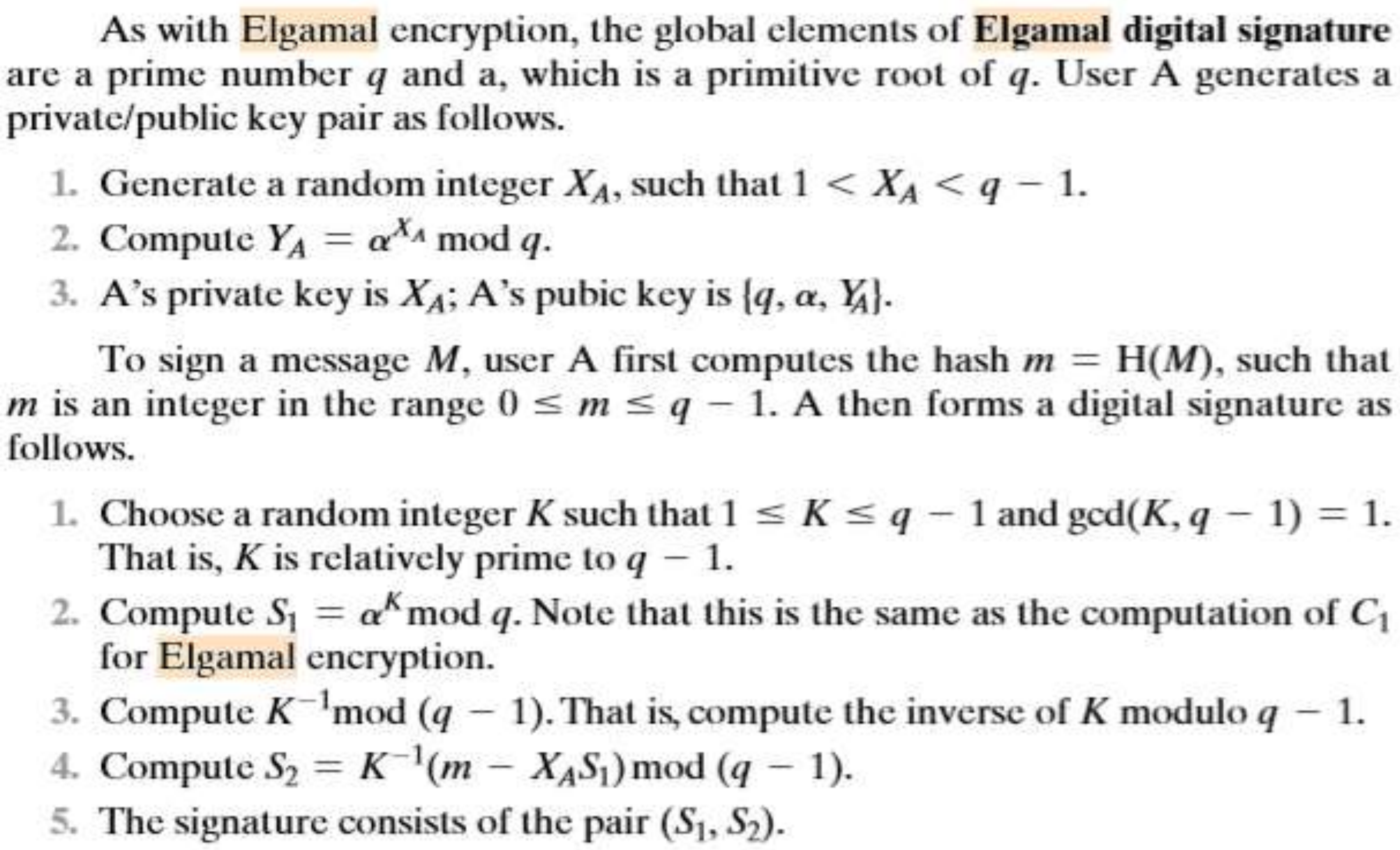
- involve only sender & receiver
- assumed receiver has sender’s public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key for confidentiality
- important that sign first then encrypt message & signature

Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

(1) $X \longrightarrow A: M \| \| E(K_{xa}, [ID_X \| \| H(M)])$
(2) $A \longrightarrow Y: E(K_{ay}, [ID_X \| \| M \| \| E(K_{xa}, [ID_X \| \| H(M)]) \| \| T])$
(a) Conventional Encryption, Arbiter Sees Message
(1) $X \longrightarrow A: ID_X \| \| E(K_{xy}, M) \| \| E(K_{xa}, [ID_X \| \| H(E(K_{xy}, M))])$
(2) $A \longrightarrow Y: E(K_{ay}, [ID_X \| \| E(K_{xy}, M)]) \| \| E(K_{xa}, [ID_X \| \| H(E(K_{xy}, M)) \| \| T])$
(b) Conventional Encryption, Arbiter Does Not See Message
(1) $X \longrightarrow A: ID_X \| \| E(PR_x, [ID_X \| \| E(PU_y, E(PR_x, M))])$
(2) $A \longrightarrow Y: E(PR_a, [ID_X \| \| E(PU_y, E(PR_x, M)) \| \| T])$
(c) Public-Key Encryption, Arbiter Does Not See Message

Elgamal Digital Signature Scheme



As with Elgamal encryption, the global elements of **Elgamal digital signature** are a prime number q and a, which is a primitive root of q. User A generates a private/public key pair as follows.

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

To sign a message M, user A first computes the hash $m = \mathrm{H}(M)$, such that m is an integer in the range $0 \leq m \leq q - 1$. A then forms a digital signature as follows.

1. Choose a random integer K such that $1 \leq K \leq q - 1$ and $\gcd(K, q - 1) = 1$. That is, K is relatively prime to $q - 1$.
2. Compute $S_1 = \alpha^K \bmod q$. Note that this is the same as the computation of C_1 for Elgamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair (S_1, S_2).

Any user B can verify the signature as follows.

1. Compute $V_1 = \alpha^m \bmod q$.
2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$.

The signature is valid if $V_1 = V_2$. Let us demonstrate that this is so. Assume that the equality is true. Then we have

$\alpha^m \bmod q = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$	assume $V_1 = V_2$
$\alpha^m \bmod q = \alpha^{X_A S_1}\alpha^{KS_2} \bmod q$	substituting for Y_A and S_1
$\alpha^{m - X_A S_1} \bmod q = \alpha^{KS_2} \bmod q$	rearranging terms
$m - X_A S_1 \equiv KS_2 \bmod (q - 1)$	property of primitive roots
$m - X_A S_1 \equiv KK^{-1}(m - X_A S_1) \bmod (q - 1)$	substituting for S_2

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}, as shown in Table 8.3. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 16$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^{16} \bmod 19 = 4$.
3. Alice's private key is 16; Alice's pubic key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

Suppose Alice wants to sign a message with hash value $m = 14$.

1. Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S_1 = \alpha^K \bmod q = 10^5 \bmod 19 = 3$ (see Table 8.3).
3. $K^{-1} \bmod (q - 1) = 5^{-1} \bmod 18 = 11$.
4. $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1) = 11(14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

Bob can verify the signature as follows.

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$.
2. $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16$.

Thus, the signature is valid.