Cryptography and System Security (CSS)

Course Code: CSC 604

Subject Incharge

Ankita Karia

Assistant Professor

Room No. 421

email: ankitakaria@sfit.ac.in

Module 3:Hashes, Message Digests and Digital Certificates

3.1

- Cryptographic Hash Functions
- Properties of secure hash function
- Various Cryptographic Hash Functions:
  - ✓ MD5,
  - ✓ SHA-1,
  - ✓ MAC, HMAC, CMAC

3.2

- Digital Certificate
- X.509, PKI

Hash Functions

1. A hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value h = H(M).
2. A "good" hash function produces output that are evenly distributed and apparently random.
3. The principal object of a hash function is **data integrity**. → **Detects changes in message**
4. A change to any bit or bits in M results, with high probability, in a change to the hash value.
5. Values returned by a hash function are called message digest or simply hash values
6. The kind of hash function needed for security applications is referred to as a **Cryptographic Hash Function.**

Cryptograhic Hash Functions

1. A cryptographic hash function is an algorithm for which it is computationally infeasible to find either

   (a) a data object that maps to a pre-specified hash result **(the one-way property)**

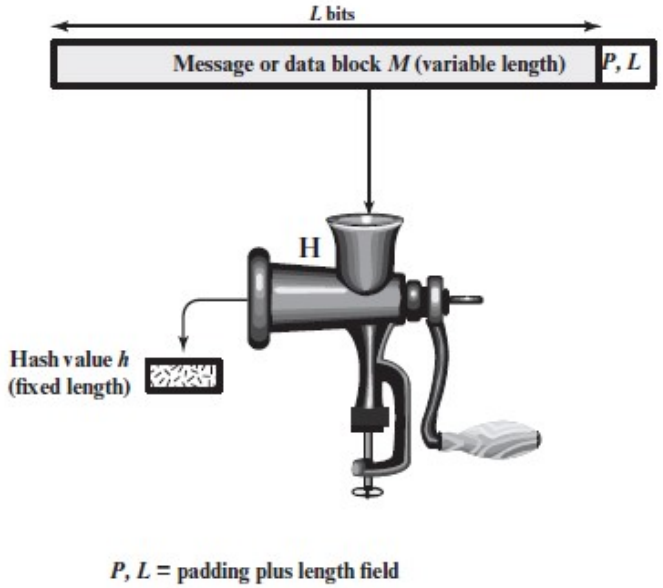
   (b) two data objects that map to the same hash result **(the collision-free property).**

2. Hash functions are often used to determine whether or not data has changed.

For a hash value h = H(x),
x is said to be the **PREIMAGE** of h.

DATA BLOCK → **HASH FUNCTION** H → **h**

Because H is a many-to-one mapping, for any given hash value h, there will in general be multiple preimages.



Cryptograhic Hash Functions - PROPERTIES

In order to be an effective cryptographic tool, the hash function is desired to possess following properties −

1. **Variable input size:** H can be applied to a block of data of any size.
2. **Fixed output size :** H produces a fixed-length output.
3. **Efficiency :** H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

A **COLLISION** occurs if we have $x \neq y$ and $H(x) = H(y)$. Because we are using hash functions for data integrity, collisions are clearly undesirable.

Cryptograhic Hash Functions - PROPERTIES

In order to be an effective cryptographic tool, the hash function is desired to possess following properties −

4. **Pre-Image Resistance (the one-way property)**
   This property means that it should be computationally hard to reverse a hash function.
5. **Second Pre-Image Resistance (weak collision resistant)**
   This property means given an input and its hash, it should be hard to find a different input with the same hash.
6. **Collision Resistance**
   This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.

A hash function that satisfies the first five properties is referred to as a **Weak Hash Function.**
If the sixth property, collision resistant, is also satisfied, then it is referred to as a **Strong Hash Function.**

The Resistant Properties Required For Various Hash Function Applications.

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

*Resistance required if attacker is able to mount a chosen message attack

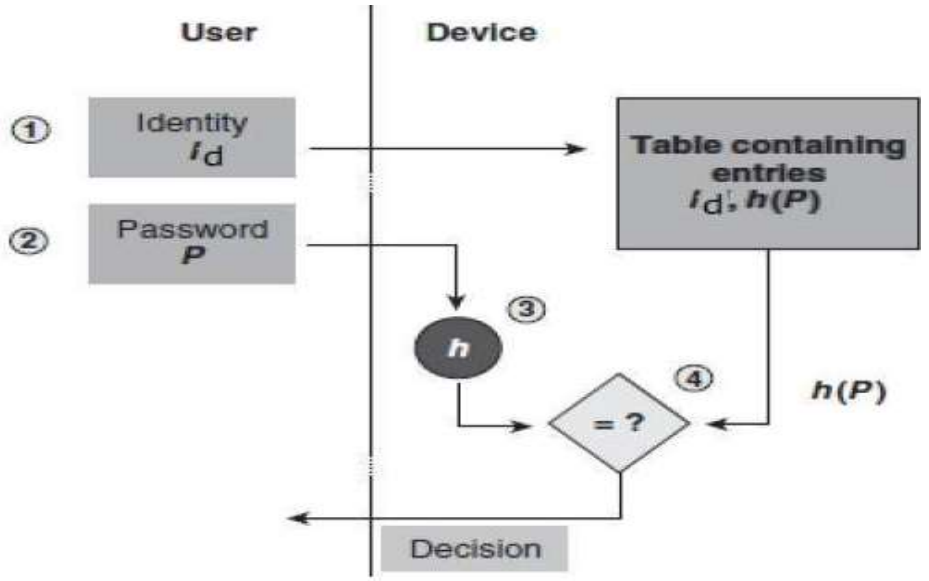
Applications of Hash Functions

There are two direct applications of hash function based on its cryptographic properties.

1. Password Storage

- ✓ All login processes store the hash values of passwords in the file.
- ✓ The Password file consists of a table of pairs which are in the form (user id, h(P)).
- ✓ The process of logon is depicted in the following illustration –

An intruder can only see the hashes of passwords, even if he accessed the password. He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.

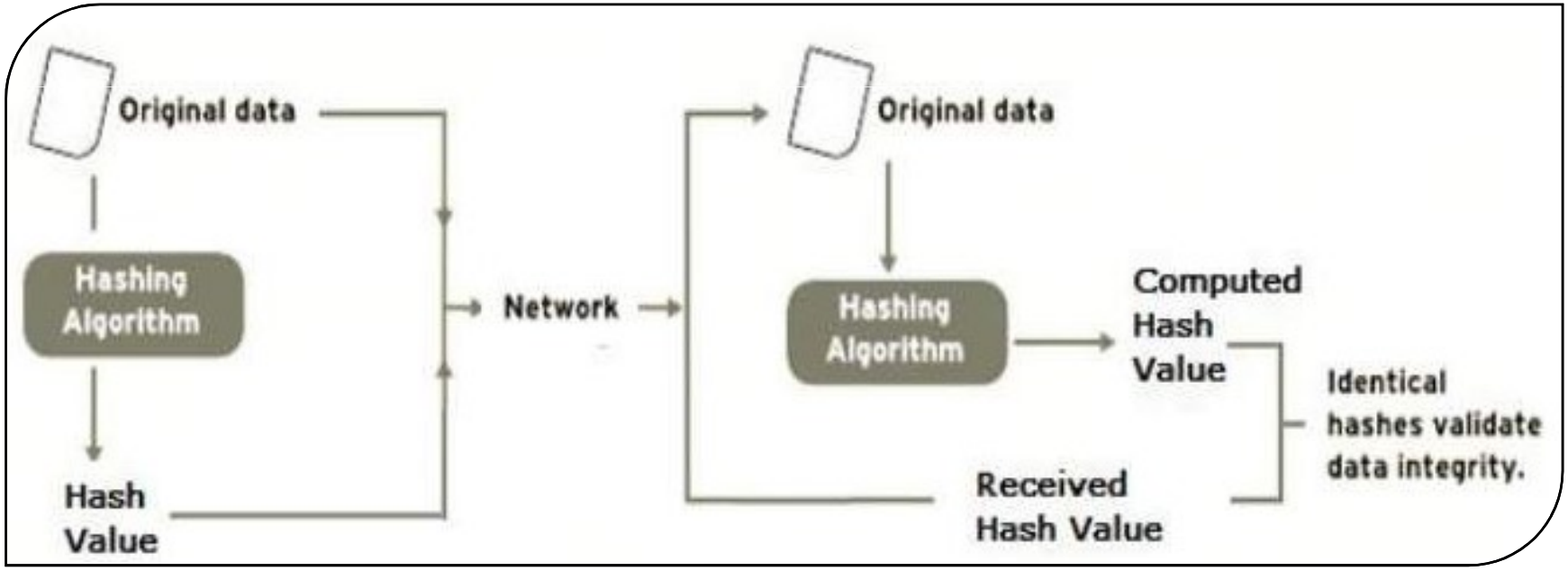


Applications of Hash Functions

There are two direct applications of hash function based on its cryptographic properties.

2. Data Integrity Check

- ✓ It is used to generate the checksums on data files.
- ✓ This application provides assurance to the user about correctness of the data.



Applications of Hash Functions

- Message Authentication:
  - Verifies the integrity of the message
- Digital signatures
- Intrusion detection and virus detection
  - keep & check hash of files on system
- Pseudorandom function (PRF) or pseudorandom number generator (PRNG)
  - PRF is used for generating symmetric keys

Various Cryptographic Hash Functions

- **Message Digest (MD)**
  - ✓ MD5 algorithm was developed by Professor Ronald L. Rivestin1991.
  - ✓ MD5 Message Digest Algorithm takes as input a message of arbitrary length and produces as output a128-bit "fingerprint“ or "message digest“ of the input.
  - ✓ MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file.
  - ✓ In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

- **Secure Hash Function (SHA)**

MD 5

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

Steps to follow:

1. **Append Padding Bits**
2. **Append Length**

After padding, **64 bits** are inserted at the end which is used to record the length of the original input

3. **Initialize MD buffer**
4. **Processing message in 16-word block**

512	- 64 = 448
1024	- 64 = 960
1536	- 64 = 1472
2048	- 64 = 1984
2560	- 64 = 2496
3072	- 64 = 3008
3584	- 64 =3520
4096	- 64 = 4032

Step 1: APPPEND PADDING BITS

Padding means adding extra bits to the original message. Padding is done such that the total bits are 64 less than a multiple of 512 bits length.

EXAMPLE:

Plaintext size = 1200 bits

Padding bits = 272

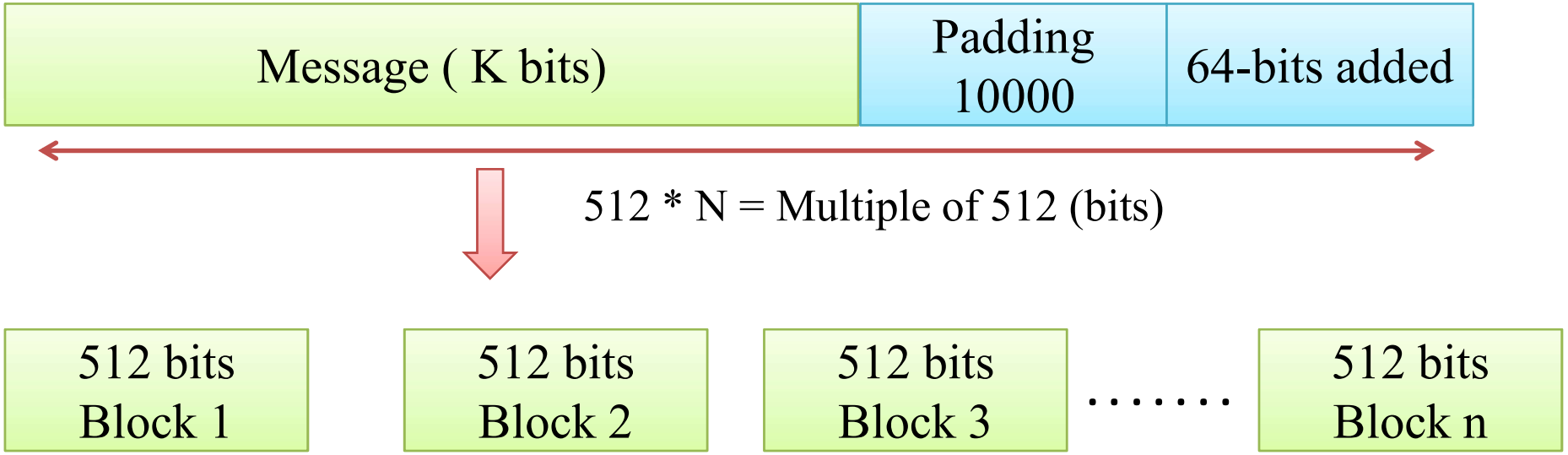
Plaintext + Padding = 1200 + 272

= 1472

MD 5

Steps to follow:

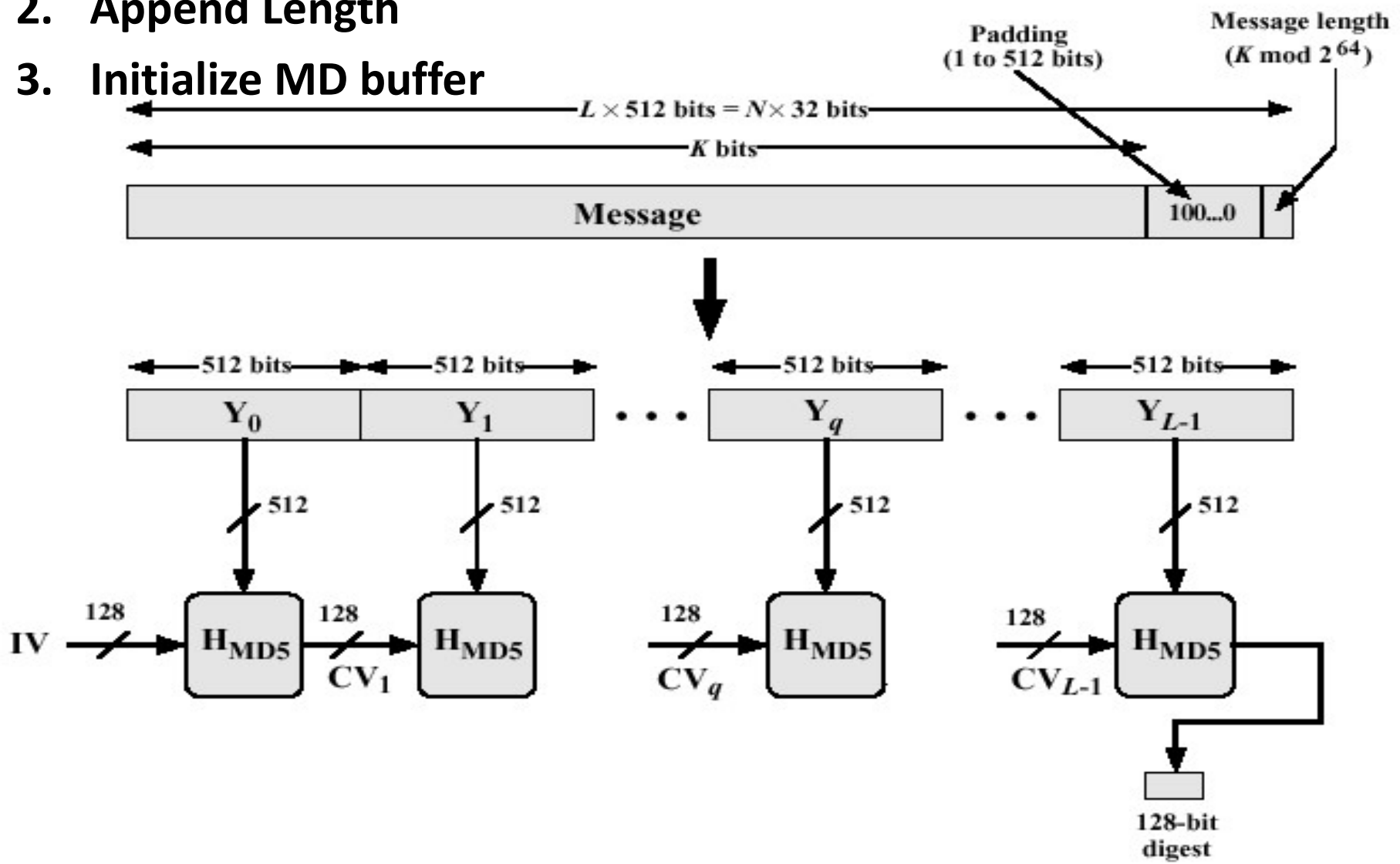
1. **Append Padding Bits**
2. **Append Length**



MD 5

Steps to follow:

1. **Append Padding Bits**
2. **Append Length**
3. **Initialize MD buffer**



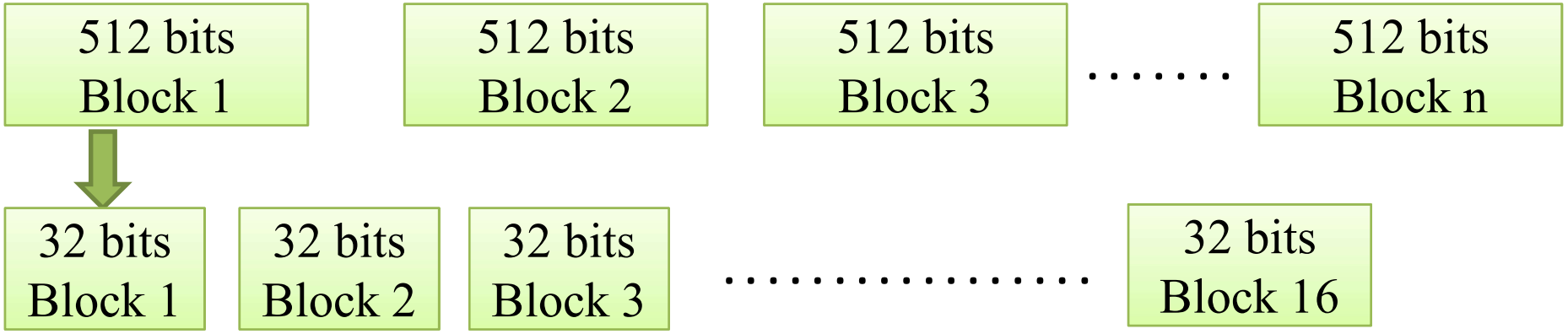
MD 5

Steps to follow:

3. Initialize MD buffer

- ✓ A 128-bit buffer is used to store intermediate and final result
- ✓ A buffer is represented as 4 32-bit register viz. A, B, C,D

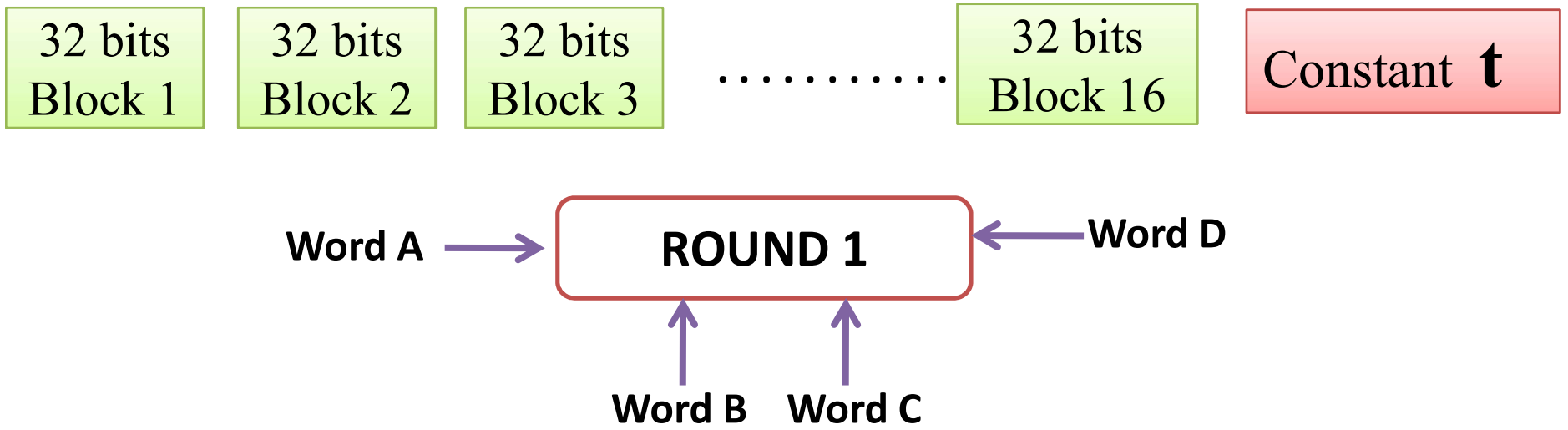
```
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10
```



MD 5

Steps to follow:

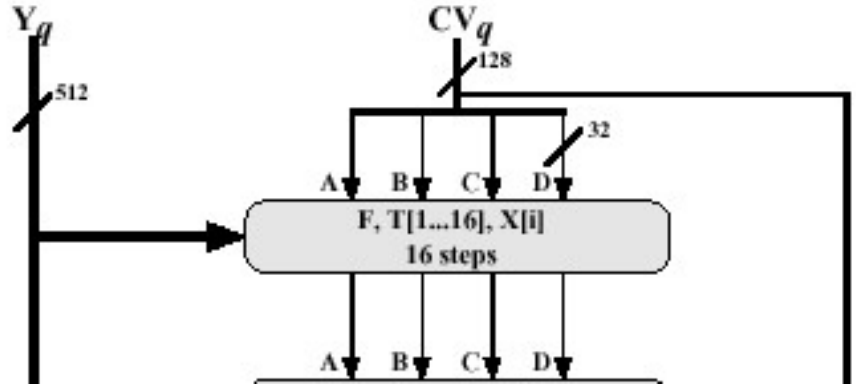
4. Processing message in 16-word block

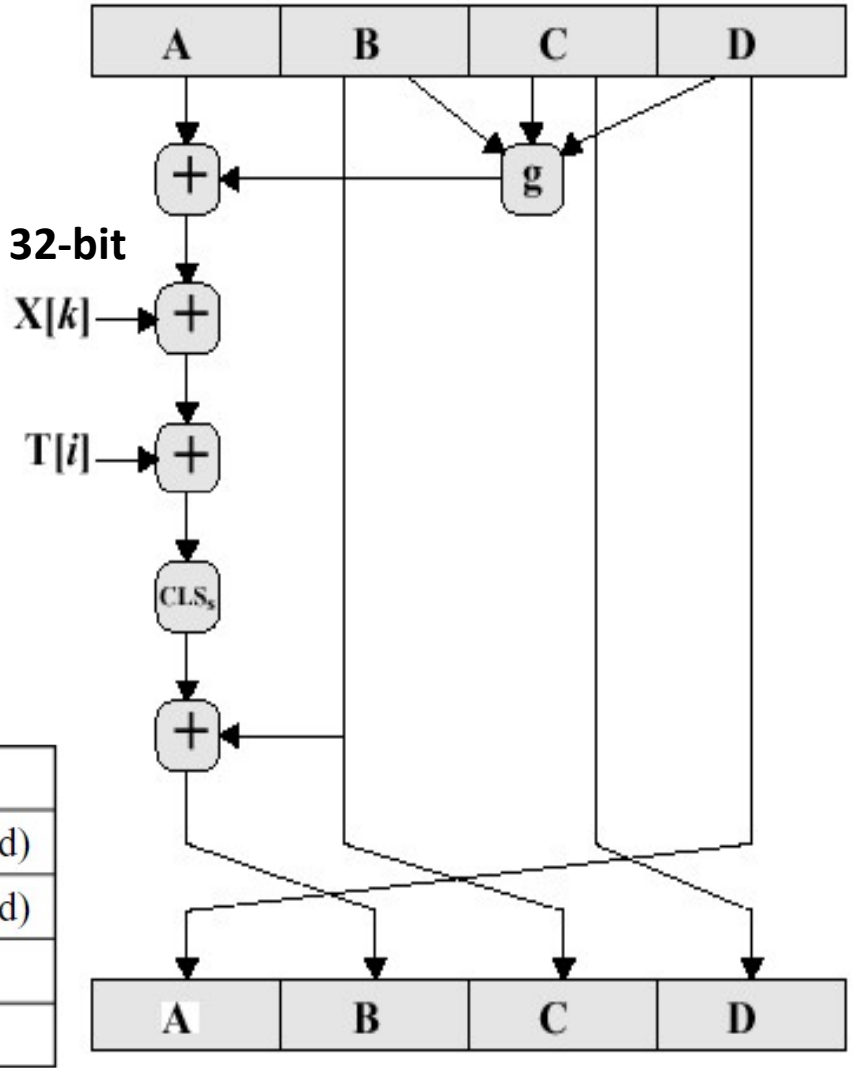


Steps to follow:

4. Processing message in 16-word block

MD 5





Round	Primitive function g	g(b,c,d)
1	F(b,c,d)	$(b \wedge c) \vee (\neg b \wedge d)$
2	G(b,c,d)	$(b \wedge d) \vee (c \wedge \neg d)$
3	H(b,c,d)	$b \oplus c \oplus d$
4	I(b,c,d)	$c \oplus (b \vee \neg d)$

Steps to follow:

4. Processing message in 16-word block

MD 5

Round	Primitive function g	g(b,c,d)
1	F(b,c,d)	$(b \wedge c) \vee (\neg b \wedge d)$
2	G(b,c,d)	$(b \wedge d) \vee (c \wedge \neg d)$
3	H(b,c,d)	$b \oplus c \oplus d$
4	I(b,c,d)	$c \oplus (b \vee \neg d)$

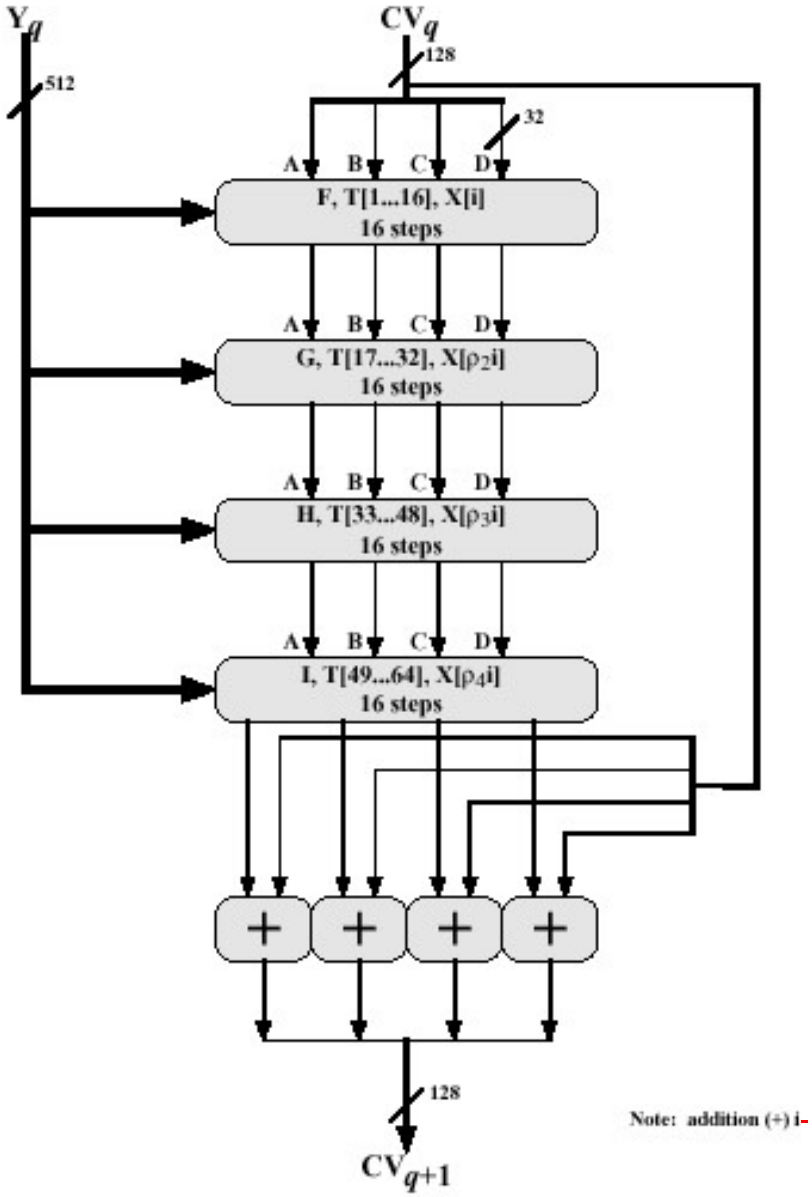


Table T, consturcted from the sine functin

T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 699D6122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57COFAF	T[21] = D62F105D	T[37] = A4BEEA44	T[53] = 655B59C3
T[6] = 4787C62A	T[22] = 02441453	T[38] = 4BDECFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBC8	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 289B7EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7BE	T[28] = 455A14ED	T[44] = 04881D05	T[60] = 4E0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D9D4D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = E6DB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB86D391

Various Cryptographic Hash Functions

- **Message Digest (MD)**
- **Secure Hash Function (SHA)**
  - ✓ The Secure Hash Algorithm (SHA) is the most widely used hash function.
  - ✓ Indeed, because virtually every other widely used hash function had been found to have substantial cryptanalytic weaknesses.
  - ✓ SHA was more or less the last remaining standardized hash algorithm by 2005.
  - ✓ SHA was developed by the National Institute of Standards and Technology (NIST)
  - ✓ When weaknesses were discovered in SHA, now known as **SHA-0**, a revised version was issued as FIPS 180-1 in 1995 and is referred to as **SHA-1.**

Versions of SHA

Secure Hash Function (SHA)

- ✓ SHA-1 produces a hash value of 160 bits.
- ✓ Later, three new versions of SHA, with hash value lengths of **256, 384, and 512** bits, known as **SHA-256, SHA-384, and SHA-512**, respectively. Collectively, these hash algorithms are known as **SHA-2.**

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

SHA-512

- Plaintext size block size – 1024 bits
- No. of Rounds/Steps – 80
- In each round a constant "k" is used
- A 512-bit buffer is used to hold intermediate and final results

8 Buffers of 64-bit each

- Size of Hash Value – 512 bits

Steps to follow:

1. Append padding bits.
2. Append length
3. Initialize hash buffer.
4. Process message in 1024-bit (128-byte) blocks.

SHA 512

Steps to follow:

1. **Append Padding Bits**
2. **Append Length**

   After padding, **128 bits** are inserted at the end which is used to record the length of the original input

3. **Initialize hash buffer**
4. **Processing message in 1024-bit block**

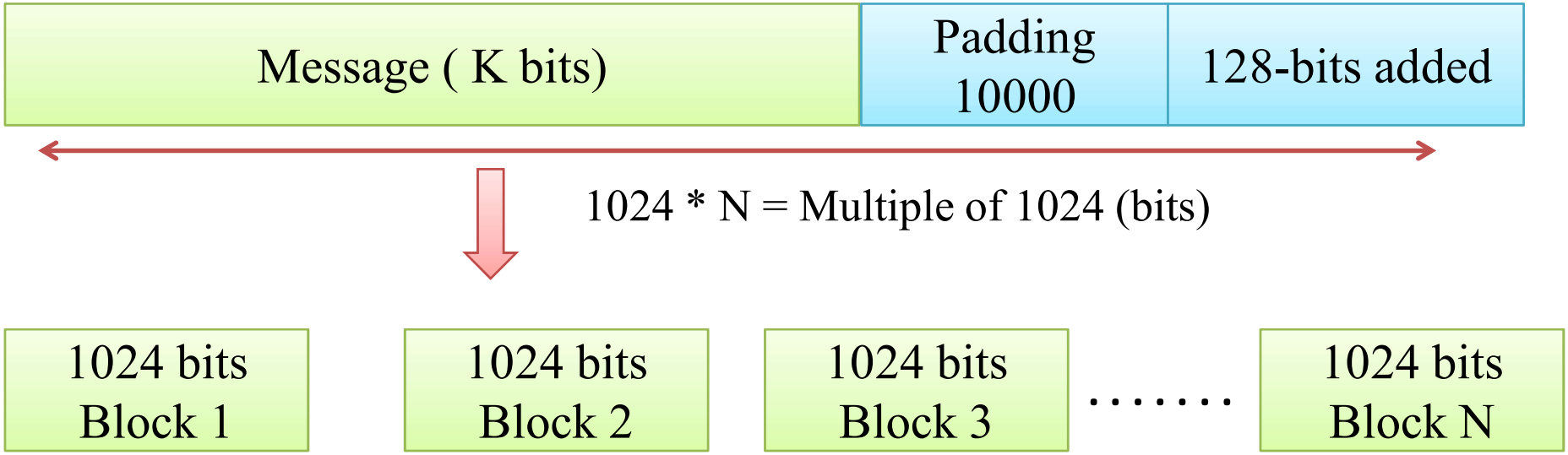
Step 1: APPPEND PADDING BITS

Padding means adding extra bits to the original message. Padding is done such that the total bits are 128 less than a multiple of 1024 bits length.

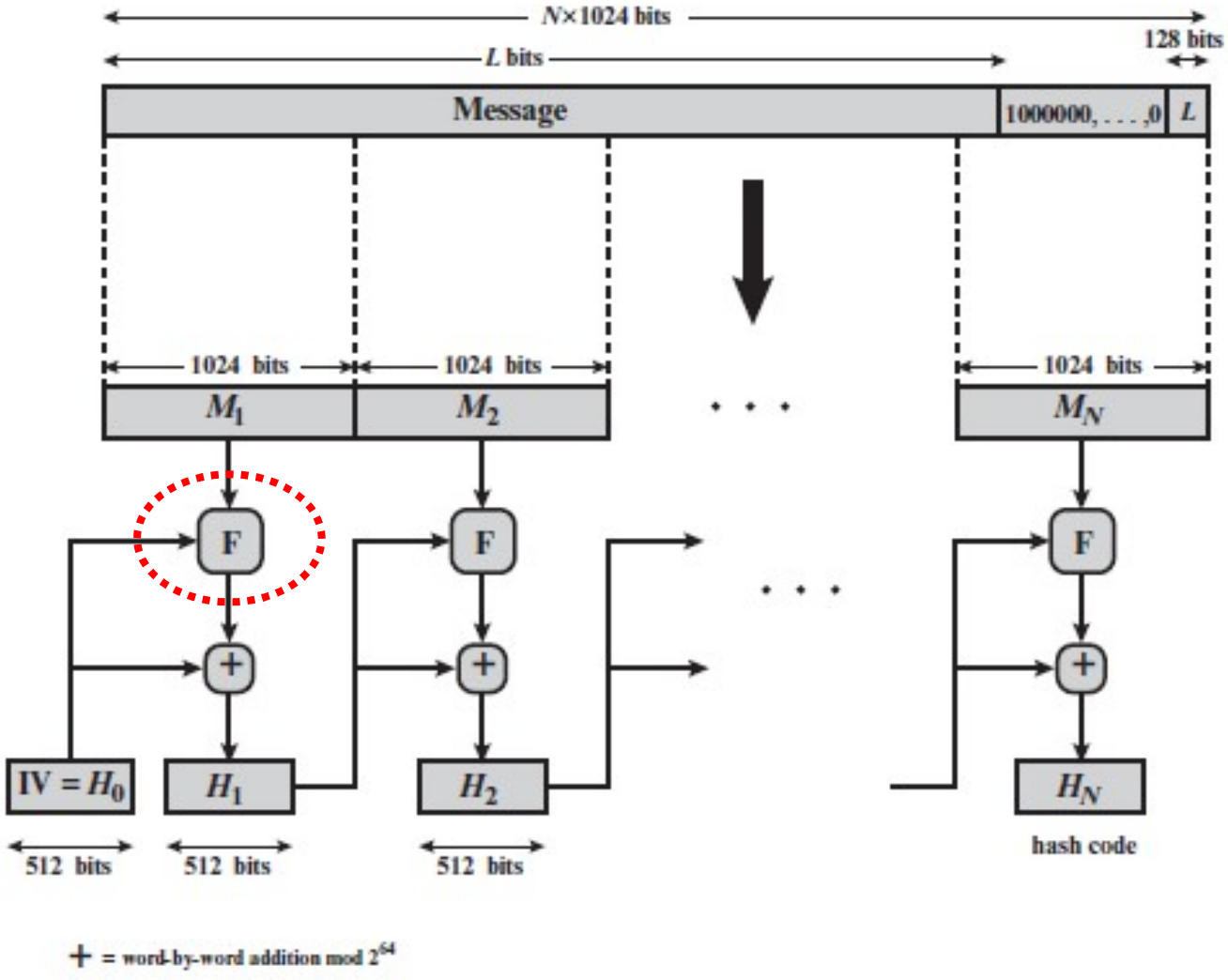
SHA 512

Steps to follow:

1. **Append Padding Bits**
2. **Append Length**



SHA 512


N×1024 bits
L bits
128 bits
Message
1000000, . . . ,0
L
1024 bits
1024 bits
1024 bits
M1
M2
MN
F
F
F
+
+
+
IV = H0
H1
H2
HN
512 bits
512 bits
512 bits
hash code
+ = word-by-word addition mod 2^64

SHA 512

Steps to follow:

3. Initialize hash buffer

- ✓ A 512-bit buffer is used to hold intermediate and final results of the hash function.
- ✓ The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).
- ✓ These registers are initialized to the following 64-bit integers (hexadecimal values)

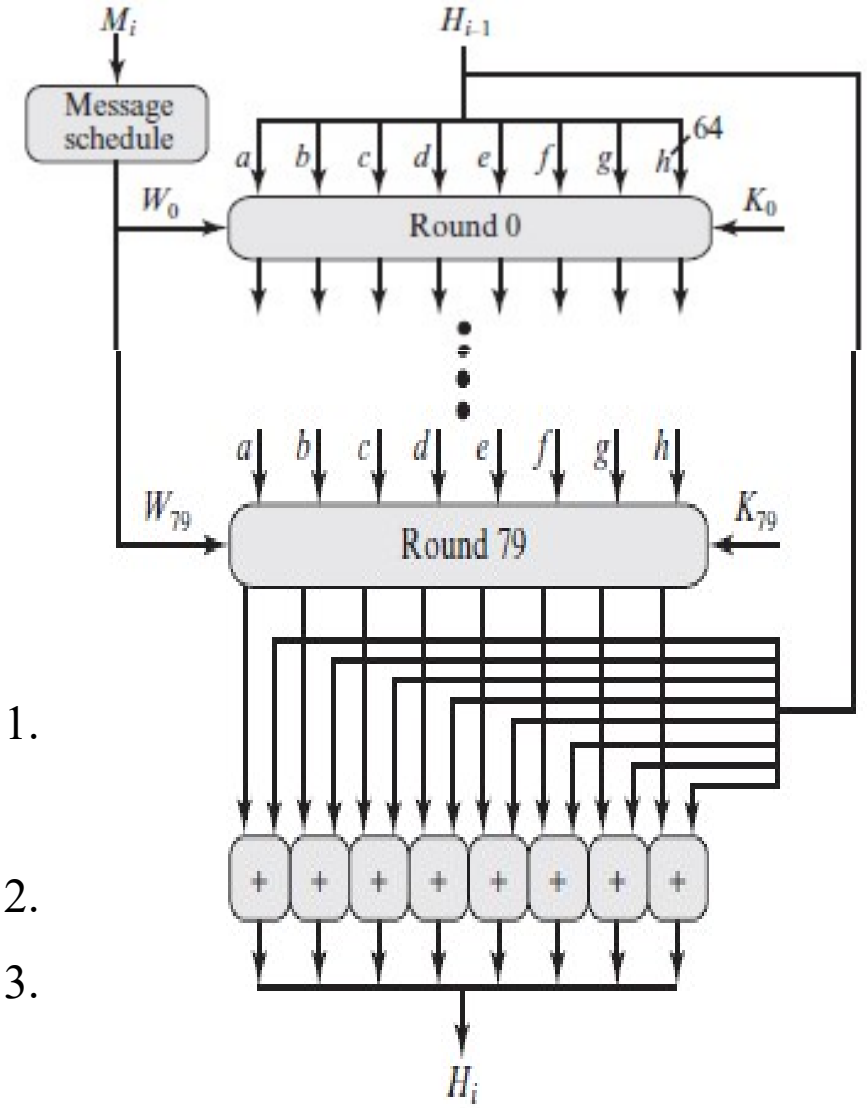
4. Process message in 1024-bit (128-byte) blocks.

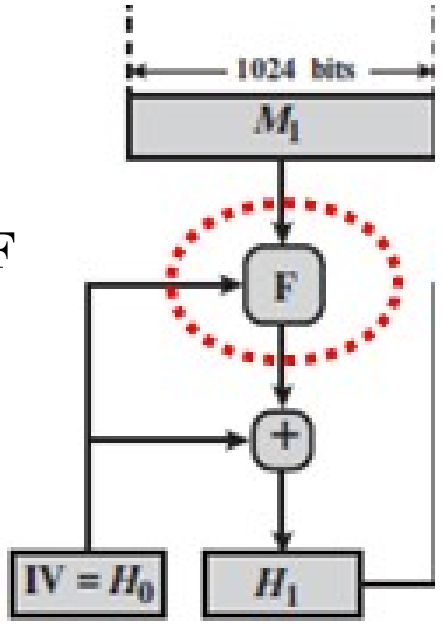
- ✓ The heart of the algorithm is a module that consists of 80 rounds.
- ✓ This module is labeled **F**

These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

SHA 512

Logic of Module F





1. ıffer value, [a b c d e f g h], and updates the
   The output of the eightieth round is added to the input to the first round (H_{i-1}) to produce H_i.
2. he value of the intermediate hash value, H_{i-1}.
3. e W_t, derived from the current 1024-bit block
4. Each round also makes use of an additive constant K_t, where 0 … t … 79 indicates one of the 80 rounds.

SHA 512

5. Output

After all N 1024-bit blocks have been processed, the output from the N^{th} stage is the 512-bit **MESSAGE DIGEST.**

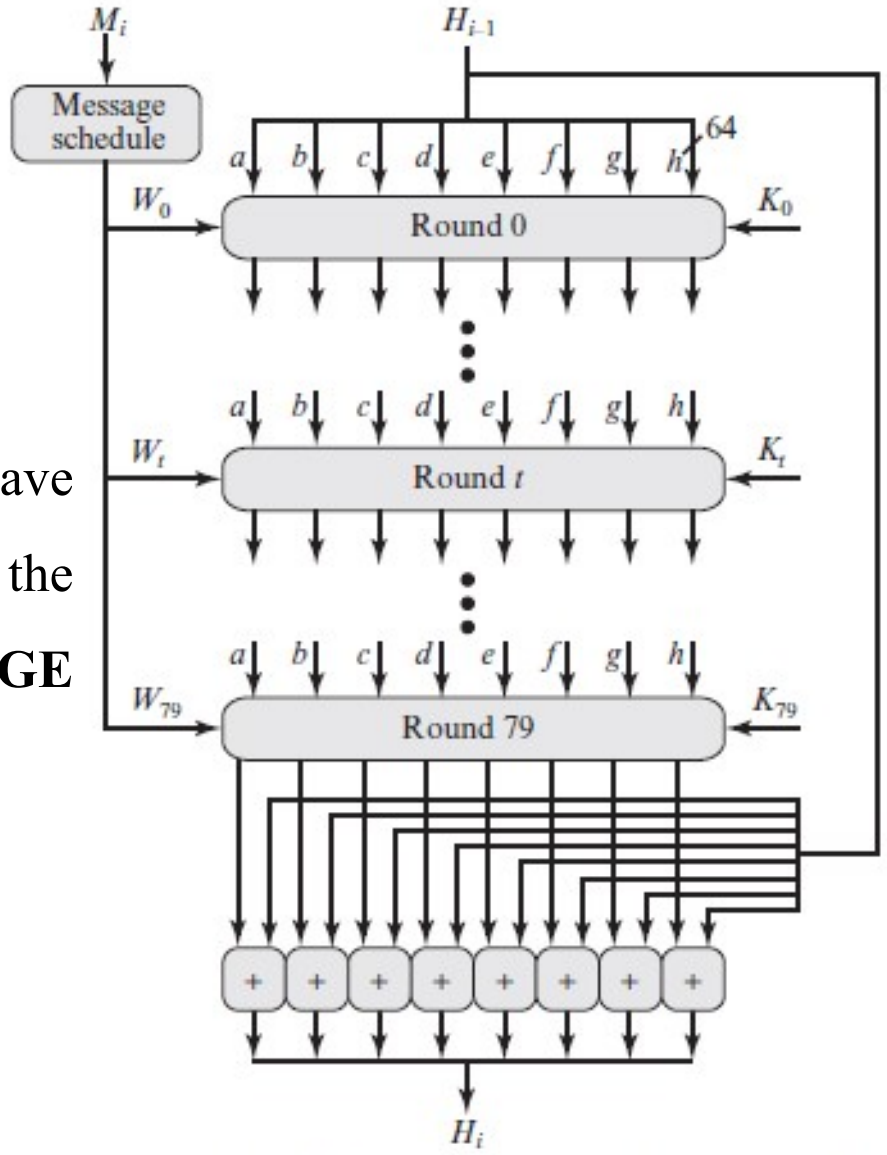


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

SHA 512 – ROUND FUNCTION

Each round is defined by the following set of equations:

$$T_1 = h + \mathrm{Ch}(e, f, g) + (\sum_1^{512} e) + W_t + K_t$$

$$T_2 = (\sum_0^{512} a) + \mathrm{Maj}(a, b, c)$$

h = g
g = f
f = e
e = d + T1
d = c
c = b
b = a
a = T1 + T2

Where;

t = step number; 0 … t … 79

Ch(e, f, g) = (e AND f) ⊕(NOT e AND g).

The conditional function: If e then f else g

Maj(a, b, c) = (a AND b) ⊕(a AND c) ⊕(b AND c): The function is true only of the majority (two or three) of the arguments are true

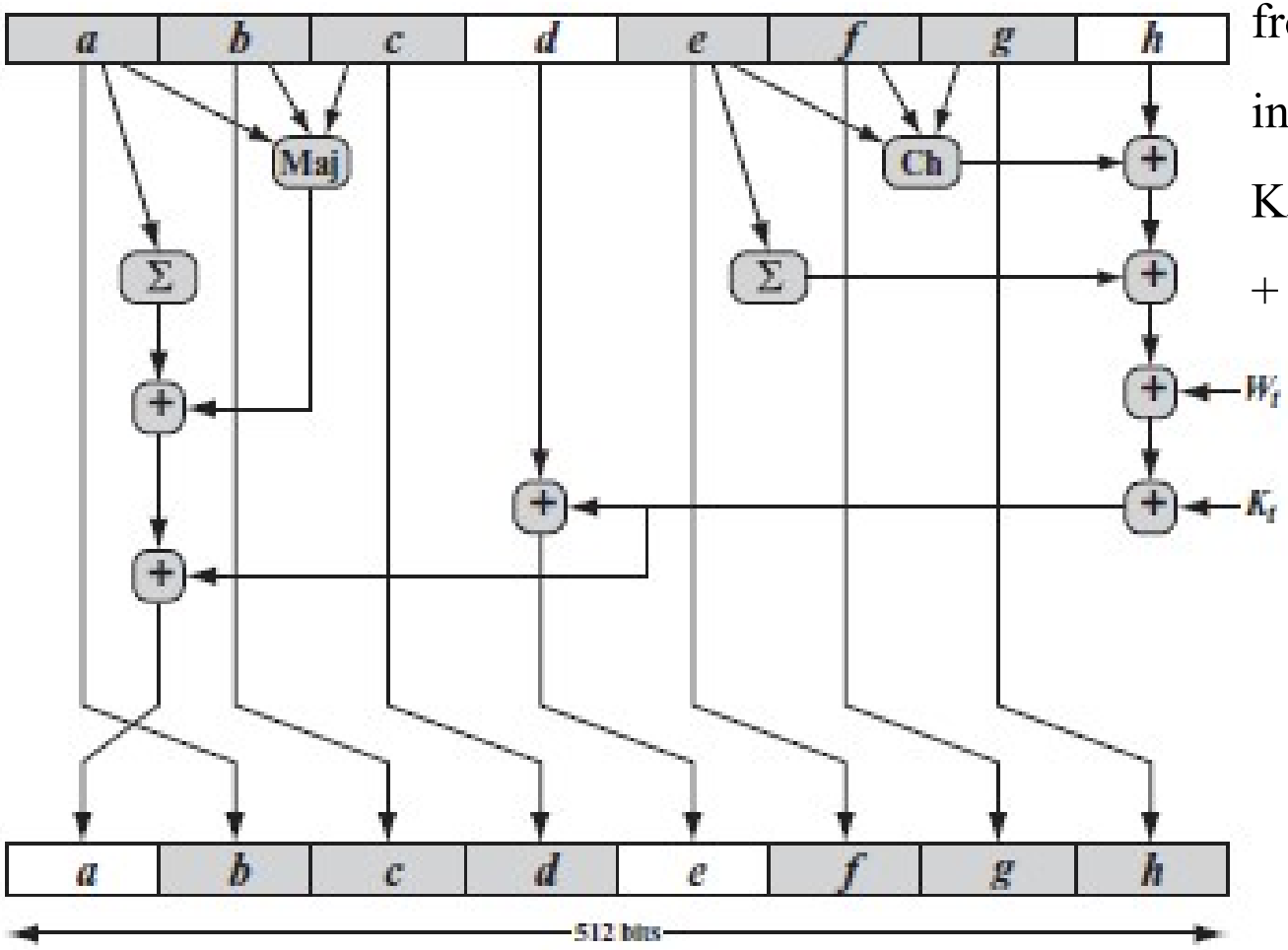
$(\sum_0^{512} a)$ = ROTR28(a) ⊕ROTR34(a) ⊕ROTR39(a)

$(\sum_0^{512} e)$ = ROTR14(e) ⊕ROTR18(e) ⊕ROTR41(e)

ROTR^n(x) = circular right shift (rotation) of the 64-bit argument x by n bits

SHA 512 – ROUND FUNCTION

Each round is defined by the following set of equations:



W_t = a 64-bit word derived from the current 1024-bit input block

K_t = a 64-bit additive constant

+ = addition modulo 2^{64}

SHA 512 – Deriving Word W_t

The first 16 values of W_t are taken directly from the 16 words of the current block.

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

where

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

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

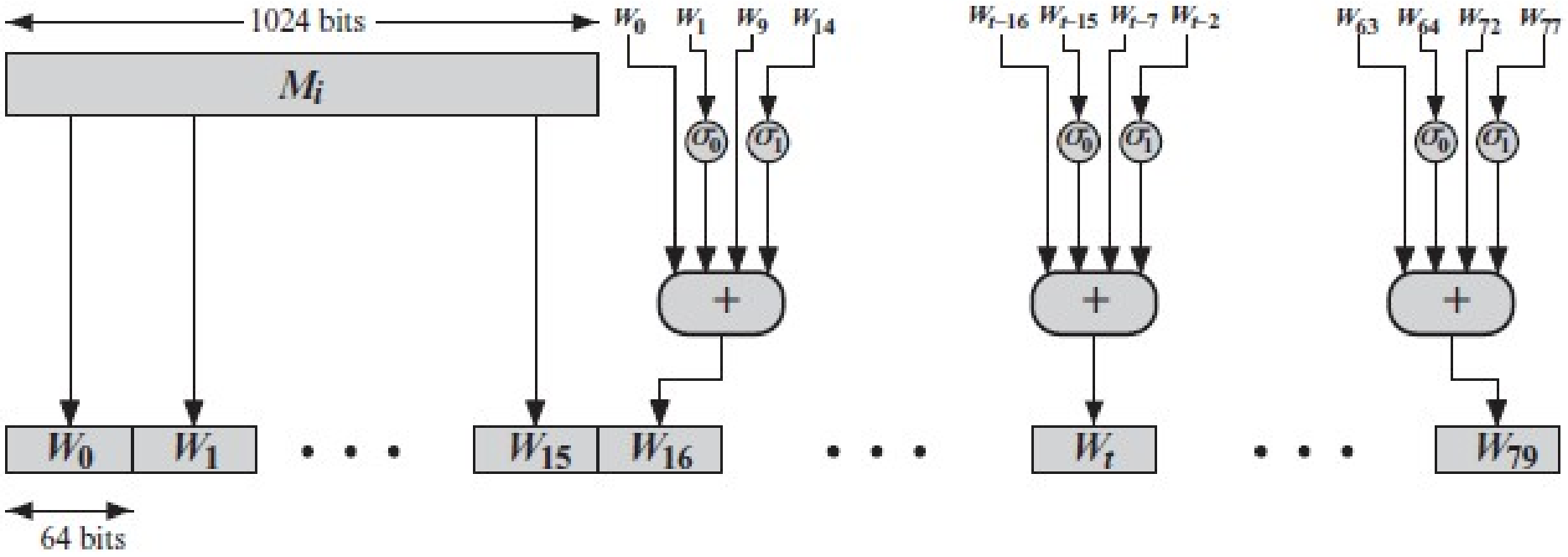
$\text{ROTR}^n(x) =$ circular right shift (rotation) of the 64-bit argument x by n bits

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

$+ =$ addition modulo 2^{64}

SHA 512 – Deriving Word W_t

The first 16 values of W_t are taken directly from the 16 words of the current block.



Module 3:Hashes, Message Digests and Digital Certificates

3.1

- Cryptographic Hash Functions
- Properties of secure hash function
- Various Cryptographic Hash Functions:
  - ✓ MD5,
  - ✓ SHA-1,
  - ✓ MAC, HMAC, CMAC

Message Authentication

1. Is a mechanism or service used to verify the integrity of a message
2. Message Authentication assures that data received are exactly as sent (i.e no Modification, Insertion, Deletion or Replay).
3. In many cases, there is a requirement that the authentication mechanism ASSURES that the purported IDENTITY of the SENDER IS VALID.

1. Message Authentication mechanism has 2 levels of functionality.

a) At lower level, there must be some sort of function that produces an authenticator → **A value to be used to Authenticate a message**

b) This is then used as a primitive in a Higher-Level authentication protocol that enables a receiver to verify the authenticity of a message

Message Authentication Code (MAC)

FUNCTIONS USED TO PRODUCE AN AUTHNETICATOR

1. HASH FUNCTION

2. MESSAGE ENCRYPTION

3. MESSAGE AUTHENTICATION CODE (MAC)

   A function of the message and a secret key that produces a FIXED-LENGTH value as the authenticator

Message Authentication Code (MAC)

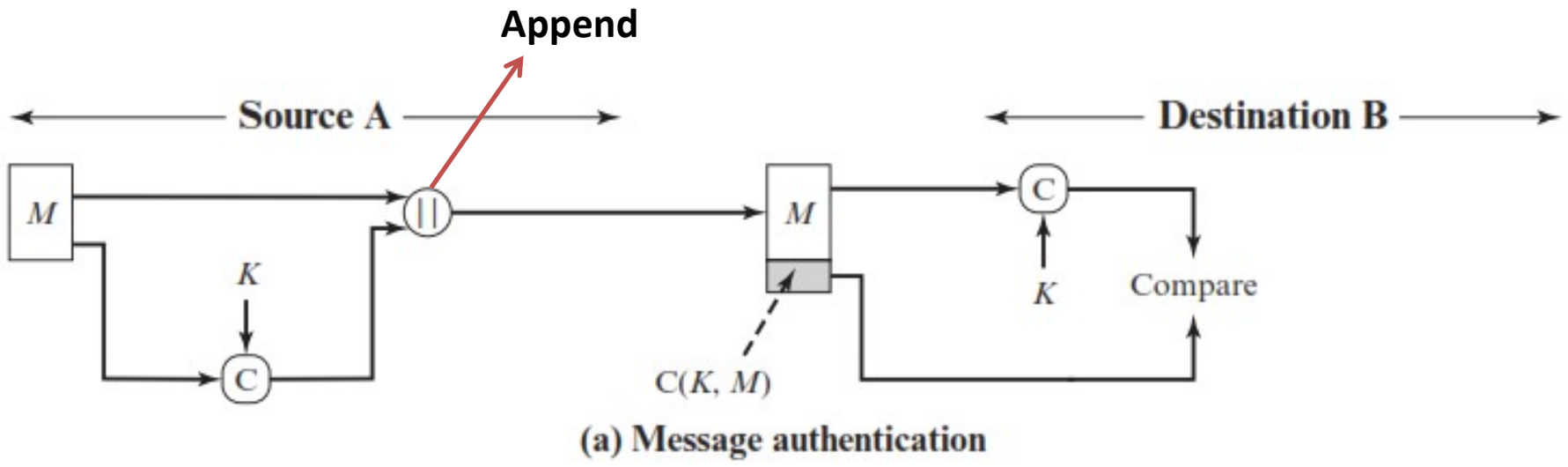
1. Involves the use of a secret key to generate a small-fixed size block of data, known as **CRYPTOGRAPHIC CHECKSUM or MAC**, that is appended to the message.
2. This technique assumes that two communicating parties, say A and B, share a common secret key K.
3. When A has a message to send to B, it calculates the MAC as a function of the message and the key:

$$\mathbf{MAC = C(}\boldsymbol{K, M}\mathbf{)}$$

where
M = *input message*
C = MAC function
K = *shared secret key*
MAC = message authentication code

Message Authentication Code (MAC)

1. The message plus MAC are transmitted to the intended recipient.
2. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.
3. The received MAC is compared to the calculated MAC



(a) Message authentication

MAC = C(*K, M*)

Message Authentication Code (MAC)

Assumption: The receiver and the sender know the identity of the secret key.

If the received MAC matches the calculated MAC, then

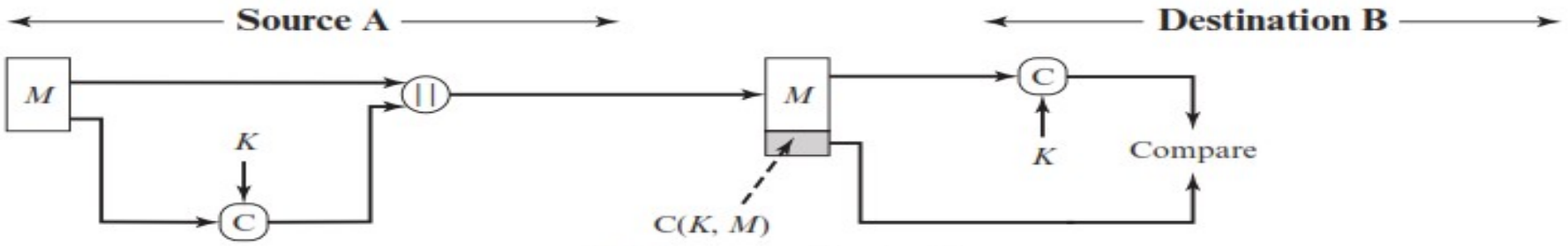
1. **The receiver is assured that the message has not been altered.**

   If an attacker alters the message but does not alter the MAC, then the receiver's calculation of The MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.

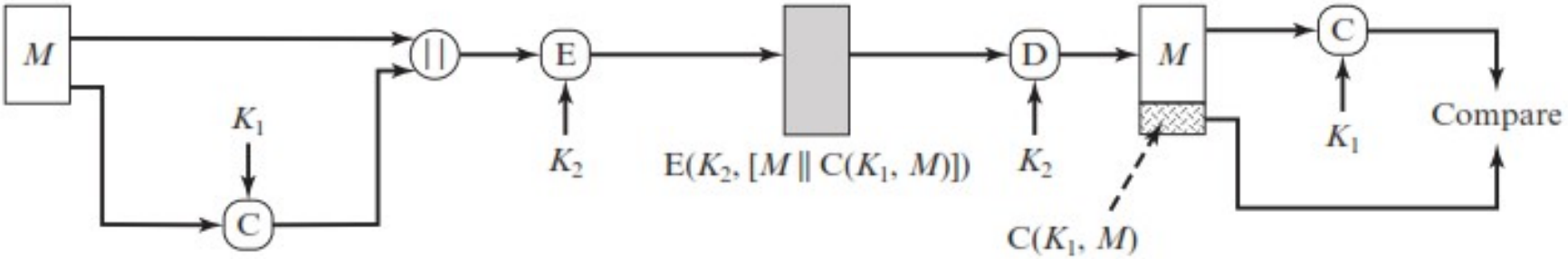
2. **The receiver is assured that the message is from the alleged sender.**

   Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

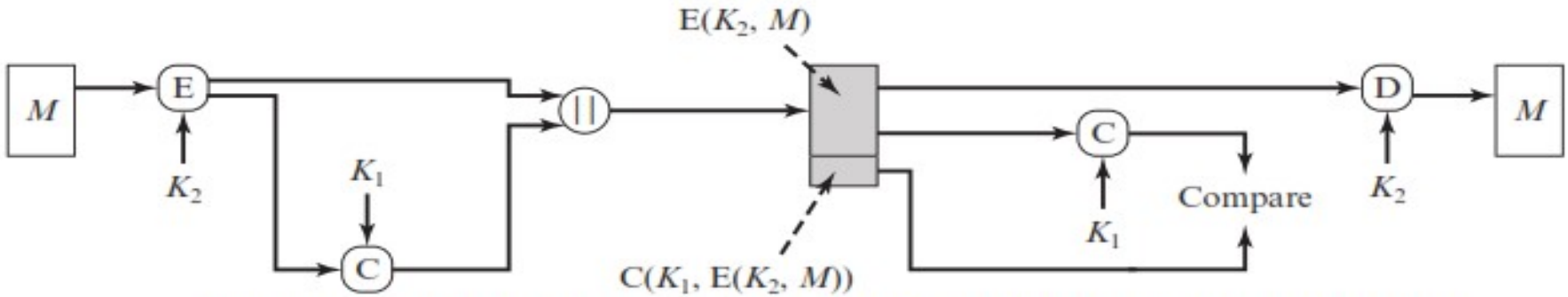
Message Authentication Code (MAC)



(a) Message authentication



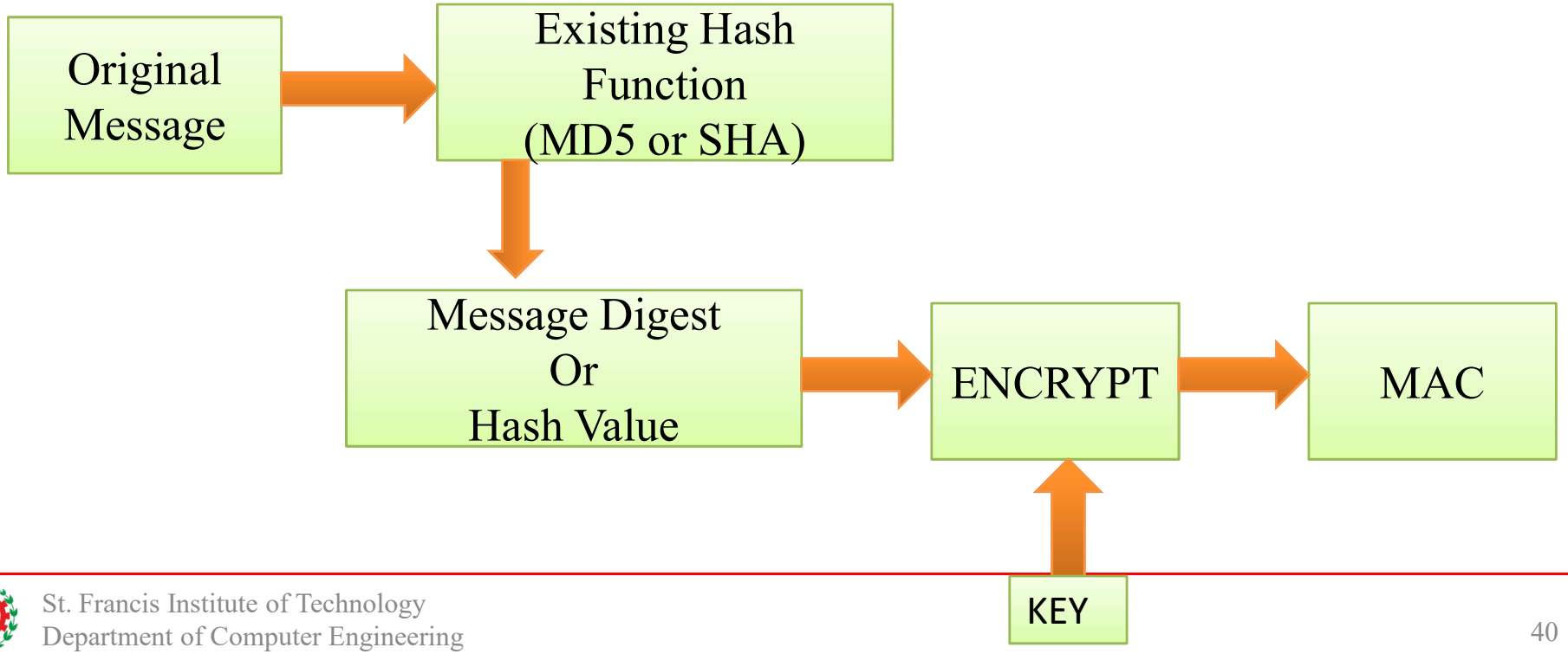
(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

MAC Based on Hash Function: HMAC

1. HMAC stands for HASH Message Authentication Code
2. It is a specific technique for calculating a Message Authentication Code (MAC) involving a combination of cryptographic hash function and a secret key cryptography.



HMAC Design Objectives

The following are design objectives for HMA

a) To use, without modifications, available hash functions.

b) To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.

c) To preserve the original performance of the hash function without incurring significant degradation

d) To use and handle keys in a simple way.

e) To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

HMAC ALGORITHM

Terms used in HMAC Algorithm:

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)

IV = initial value input to hash function

M = message input to HMAC (including the padding specified in the embedded hash function)

Y= i^{th} block of M, $0 \leq i \leq (L - 1)$

L = number of blocks in M

b = number of bits in a block

HMAC ALGORITHM

Terms used in HMAC Algorithm:

n = length of hash code produced by embedded hash function

K = secret key; recommended length is ≥ n; if key length is greater than b, the key is input to the hash function to produce an n-bit key

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

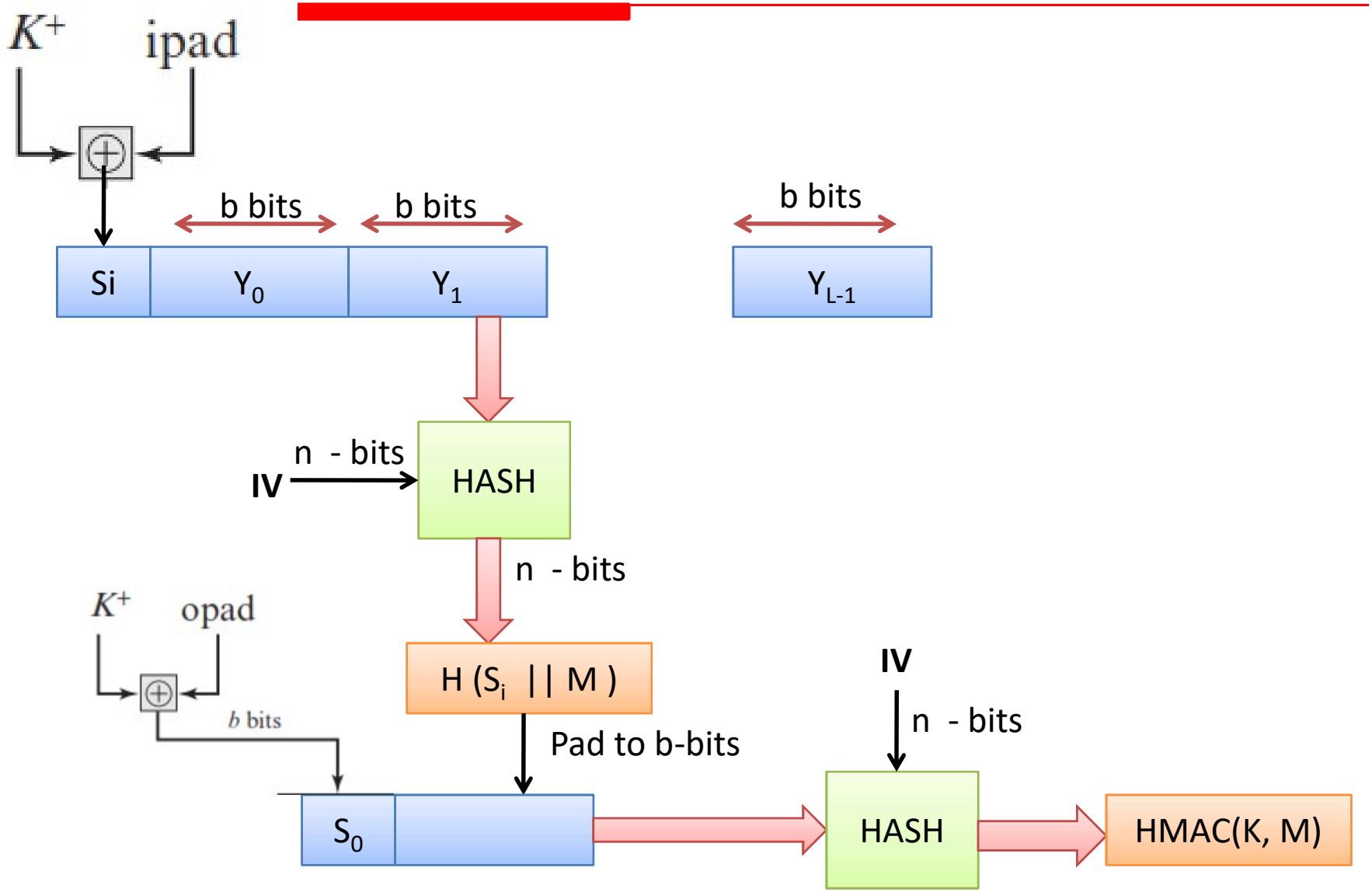
ipad = 00110110 (36 in hexadecimal) repeated b/8 times

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

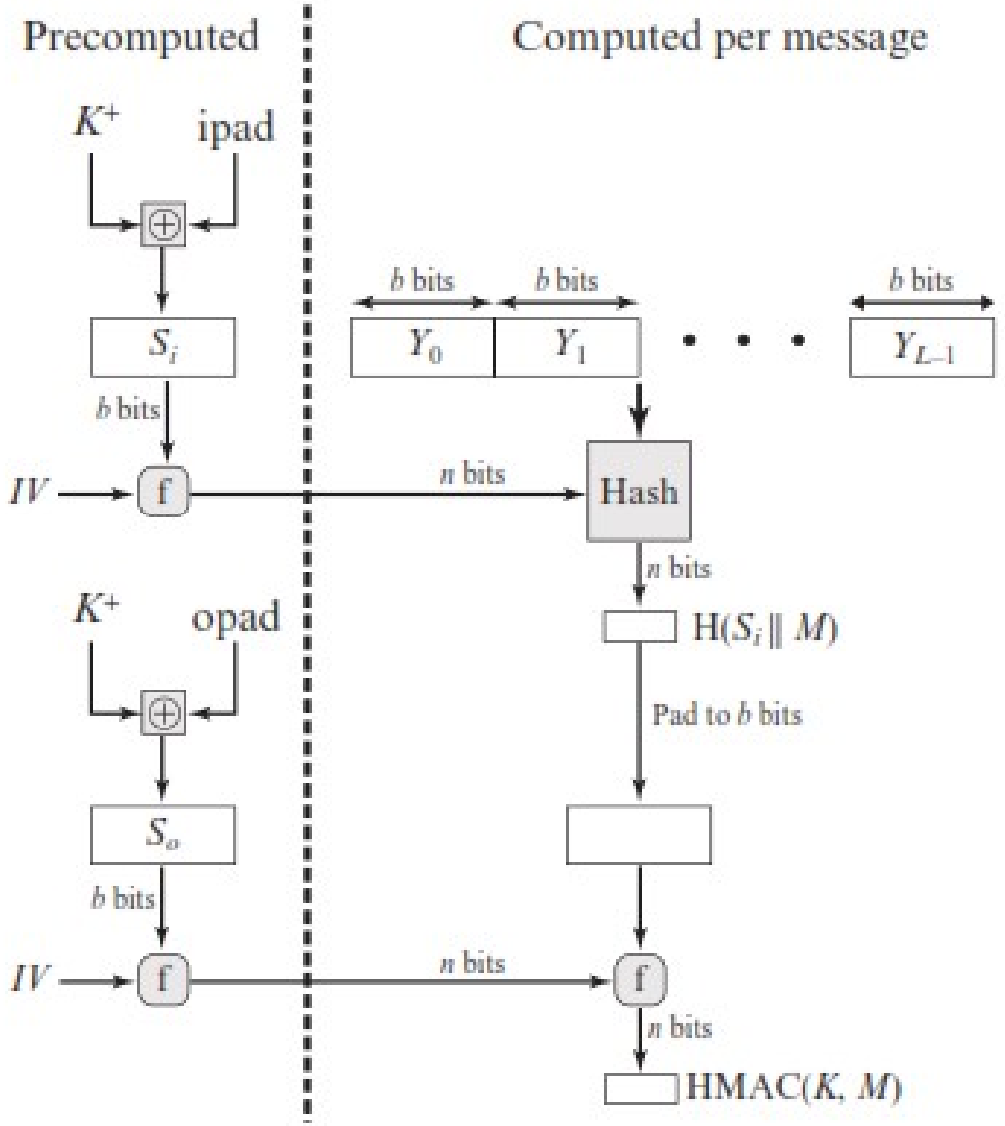
HMAC can be expressed as

$$HMAC(K, M) = H[(K^{+} \oplus opad) \| H[(K^{+} \oplus ipad) \| M]]$$

HMAC STRUCTURE



HMAC STRUCTURE



CMAC : Cipher based Message Authentication Code

1. Cipher-based message authentication codes (or CMACs) are a tool for calculating message authentication codes using a block cipher coupled with a secret key.
2. CMAC is used to verify both the integrity and authenticity of a message.
3. It is a mode of operation for use with AES and Triple DES

Operation of CMAC:

- For AES, the cipher block length, b = 128 bits
- For Triple DES, the cipher block length, b = 64 bits
- The message is divided into n blocks (M_1, M_2, M_3, ……., M_n)
- k-bit encryption key K is used **Triple DES key size is 112 or 168 bits**
  AES key size is 128, 192, 256 bits
- A constant K1 is used which is length b-bits

Message is integer multiple of cipher block length

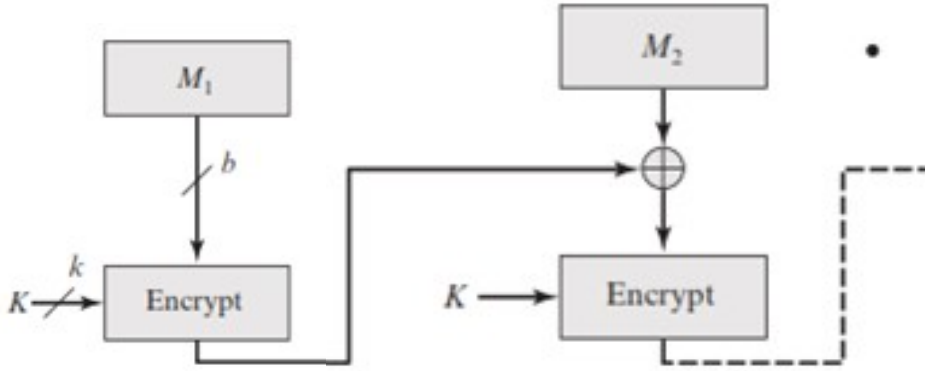
CMAC : Cipher based Message Authentication Code

CMAC is calculated as follows:

$$C_1 = E(K, M_1)]$$

$$C_2 = E(K, [M_2 \oplus C_1)$$

$$C_3 = E(K, [M_3 \oplus C_2)$$



$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1 \quad)$$

$$T = MSB_{Tlen}(C_n)$$

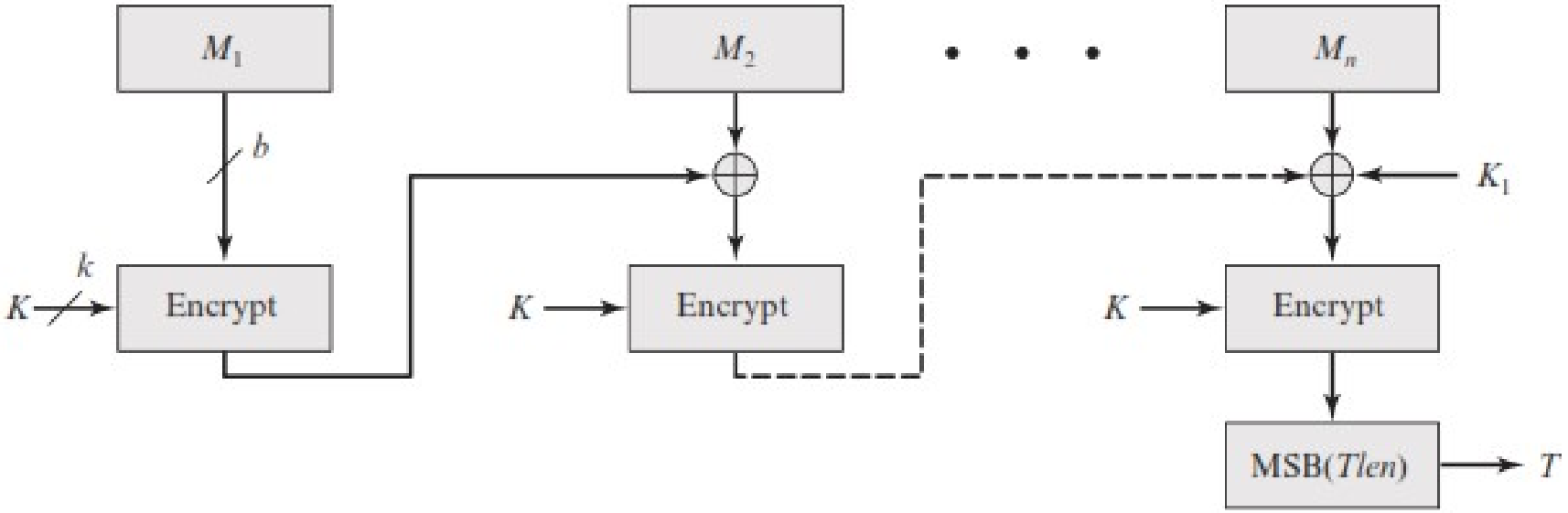
Where,

T = MAC, also referred to as the TAG.

Tlen = bit length of T

MSBs(X) = the s leftmost bits of the bit string X

CMAC : Cipher based Message Authentication Code

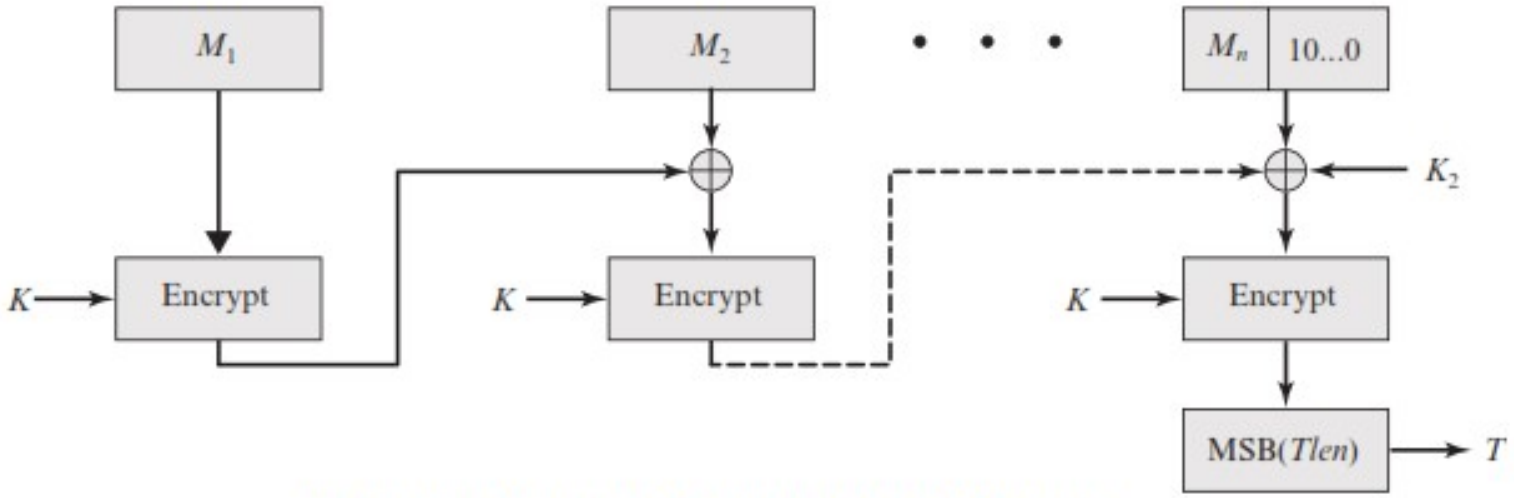


(a) Message length is integer multiple of block size

CMAC : Cipher based Message Authentication Code

The final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b.

The CMAC operation then proceeds as before, except that a different b-bit key K2 is used instead of K1



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

CMAC : Cipher based Message Authentication Code

The two b-bit keys are derived from the k-bit encryption key as follows:

$$L = \mathrm{E}(K, 0^b)$$
$$K_1 = L \cdot x$$
$$K_2 = L \cdot x^2 = (L \cdot x) \cdot x$$

1. To generate K1 and K2 , the block cipher is applied to the block that consists entirely of 0 bits.
2. The first subkey is derived from the resulting ciphertext by a left shift of one bit and, conditionally, by XORing a constant that depends on the block size.
3. The second subkey is derived in the same manner from the first subkey.

Module 3:Hashes, Message Digests and Digital Certificates

3.1

- Cryptographic Hash Functions
- Properties of secure hash function
- Various Cryptographic Hash Functions:
  - ✓ MD5,
  - ✓ SHA-1,
  - ✓ MAC, HMAC, CMAC

3.2

- Digital Certificate
- X.509, PKI

Distribution of Public Key

Private key
(Sender and Receiver shares the key)
→ Symmetric Key Cryptography → Diffie-Hellman Key Exchange Algorithm

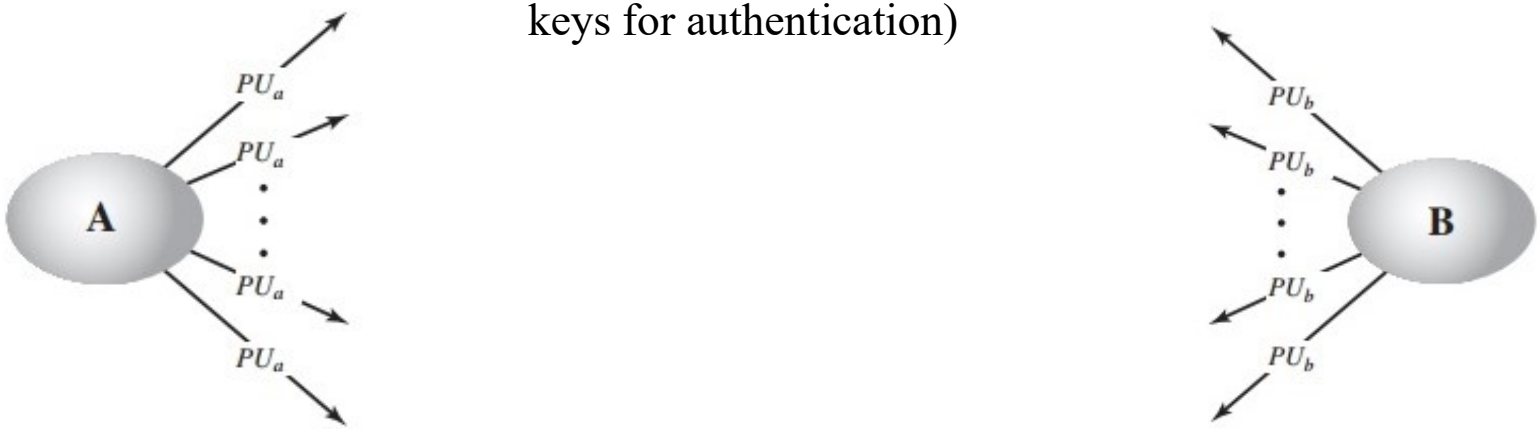
Private key
Public key
→ Asymmetric Key Cryptography →

1. Public announcement
2. Publicly available directory
3. Public-key authority
4. Public-key certificates

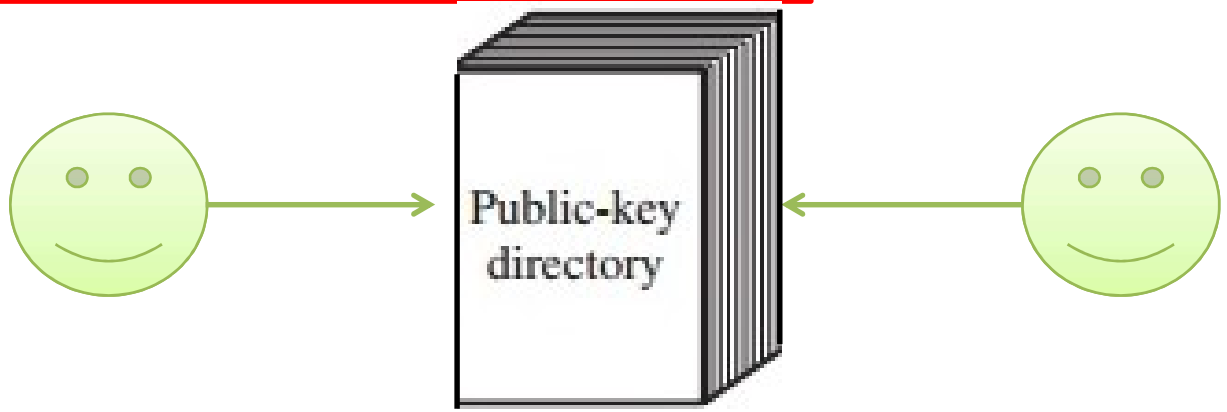
1. Public announcement

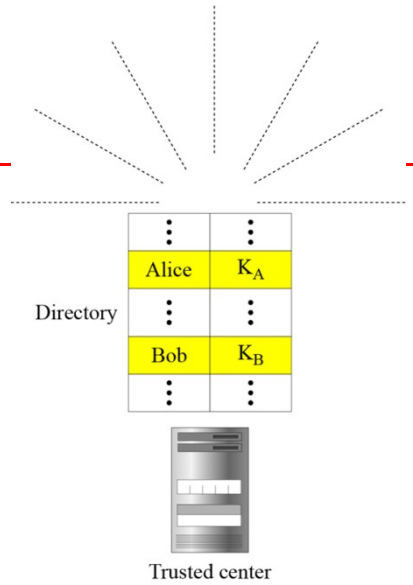
- Any participant can send his or her public key to any other participant or broadcast the key to the community at large.
- Although this approach is convenient, it has a major weakness.
- Anyone can forge such a public announcement.

(That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication)



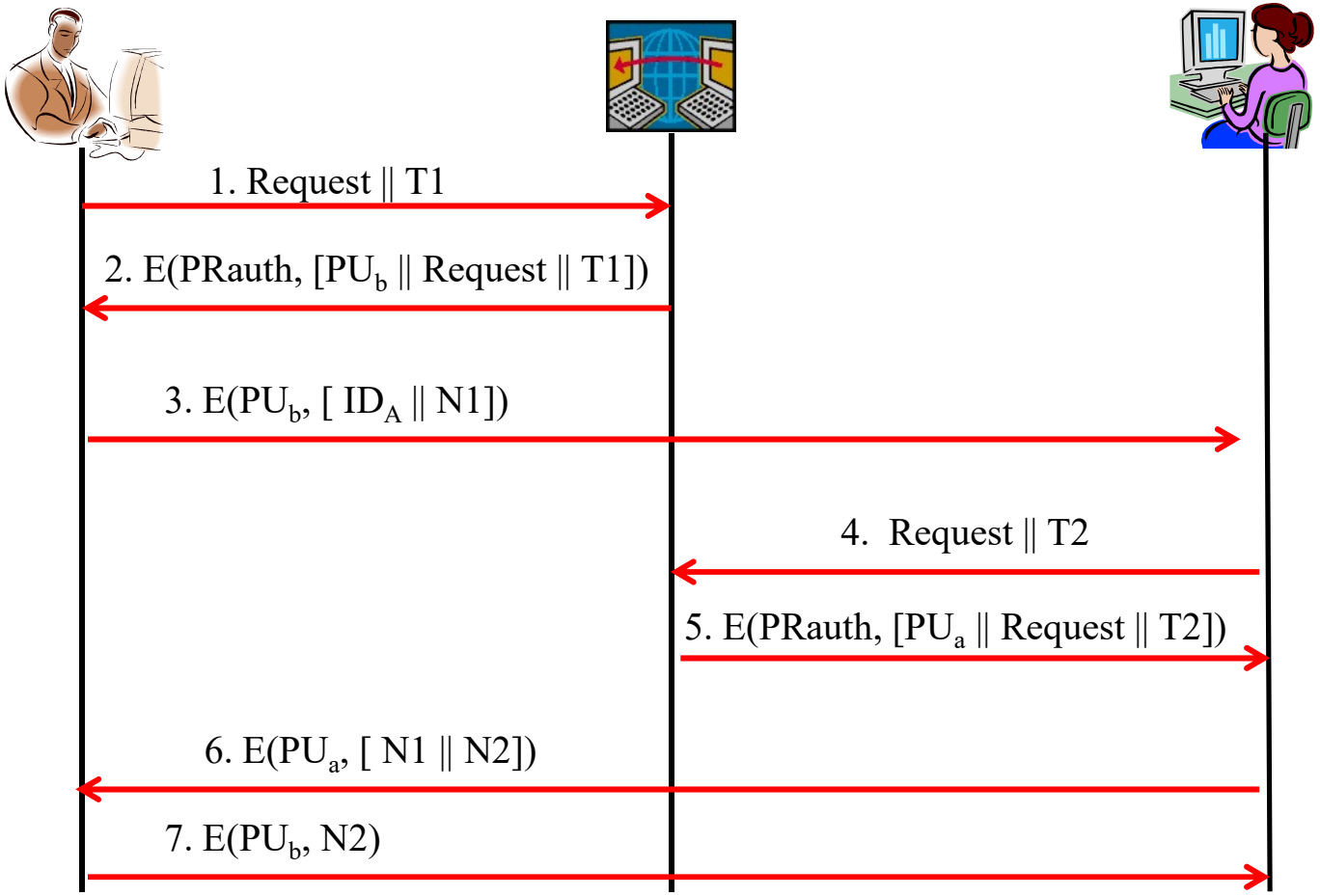
2. Publicly Available Directory





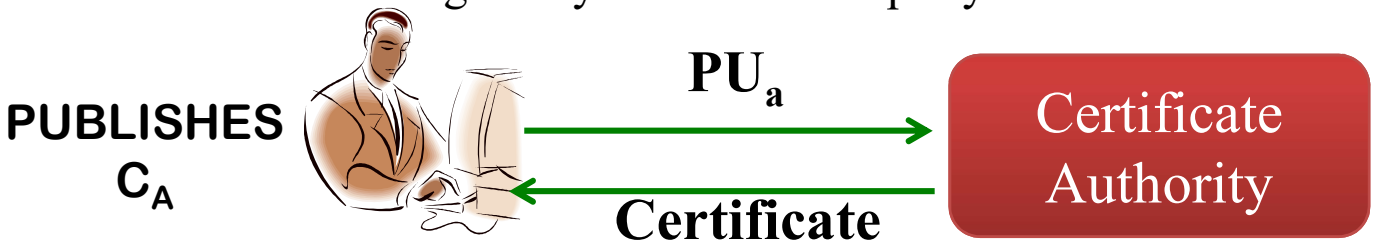
1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time,
   - ✓ A public key has already been used for a large amount of data,
   - ✓ The corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

3. Public-Key Authority



4. Public-key Certificates

1. An alternative approach, first suggested by Kohnfelder is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in such a way that is as reliable as if the keys were obtained directly from a public-key authority.
2. A certificate consists of a
   - public key,
   - an identifier of the key owner,
   - The whole block signed by a trusted third party.



$C_A = E(PRauth, [T1 || IDA || PUa])$

The third party is a certificate authority, such as a **GOVERNMENT AGENCY** or a **FINANCIAL INSTITUTION**, that is trusted by the user community.

4. Public-key Certificates

4. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.
5. A participant can also convey its key information to another by transmitting its certificate.
6. Other participants can verify that the certificate was created by the authority.

REQUIREMENTS

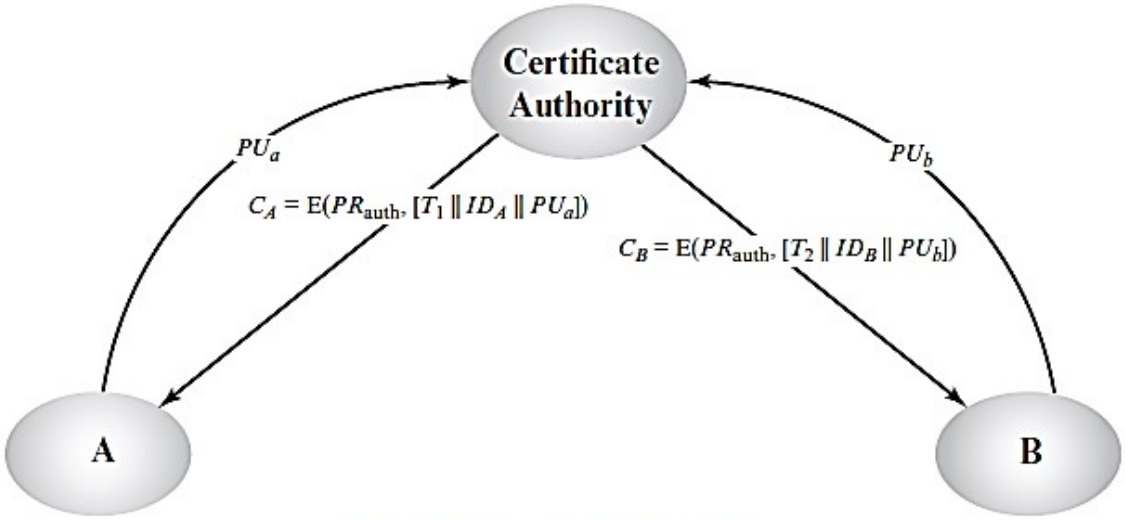
1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the time validity of the certificate.

4. Public-key Certificates

1. Each participant applies to the certificate authority, supplying a public key and requesting a certificate.
2. For participant A, the authority provides a certificate of the form

$$C_A = E(PR_{auth}, [T \| ID_A \| PUa])$$

where,
PR_{auth} is the private key used by the authority
T is a timestamp.



(a) Obtaining certificates from CA

4. Public-key Certificates

1. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

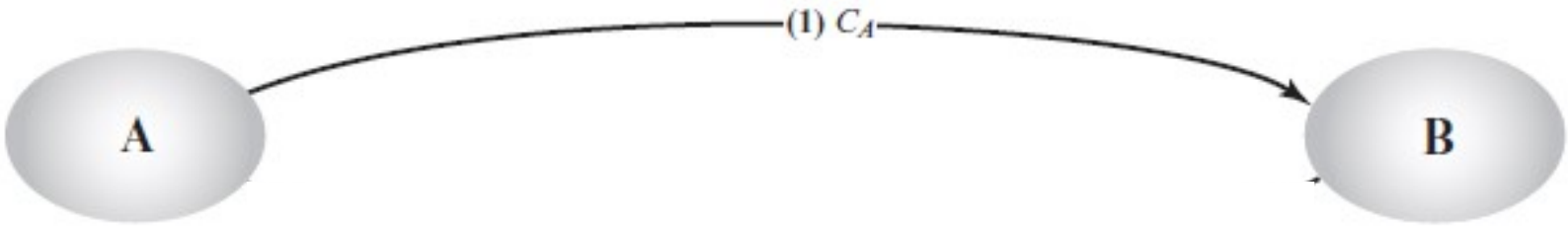
$$D(PU_{auth}, CA) = D(PU_{auth}, \quad E(PR_{auth}, [T \| ID_A \} PU_a]))$$

$$= (T \| ID_A \| PU_a)$$

The recipient uses the authority's public key, PU_{auth}, to decrypt the certificate.

The elements ID_A and PU_a provide the recipient with the name and public key of the certificate's holder.

The timestamp T validates the currency of the certificate.



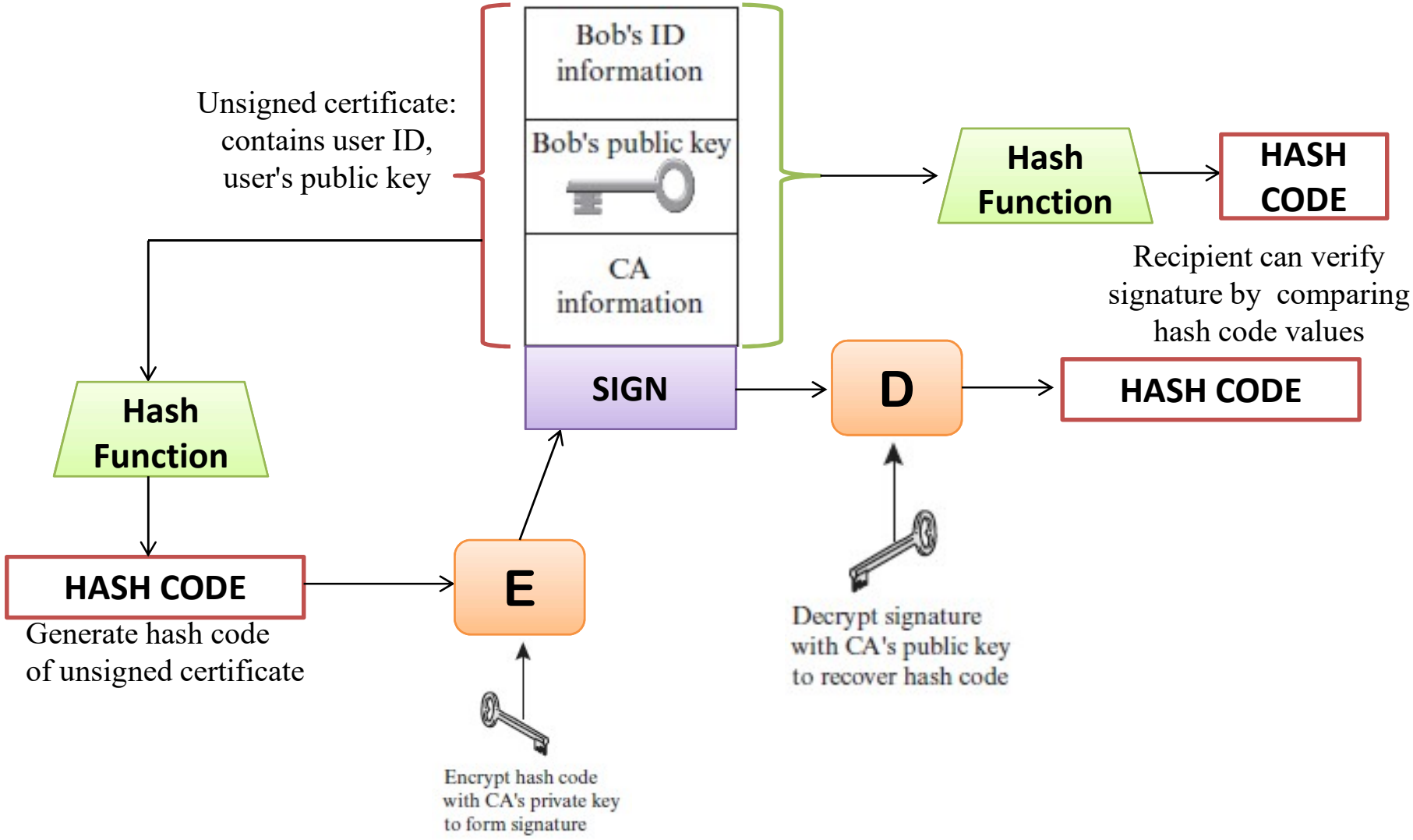
One scheme has become universally accepted for formatting public-key certificates: the **X.509 standard**.

These are used in most network security applications, including IP security, transport layer security (TLS)

X. 509 Certificates

1. X.509 was initially issued in 1988. The standard was subsequently revised in 1993 to address some of the security concerns. The standard is currently at version 7, issued in 2012.
2. X.509 is based on the use of public-key cryptography and digital signatures.
3. The standard does not dictate the use of a specific digital signature algorithm nor a specific hash function.

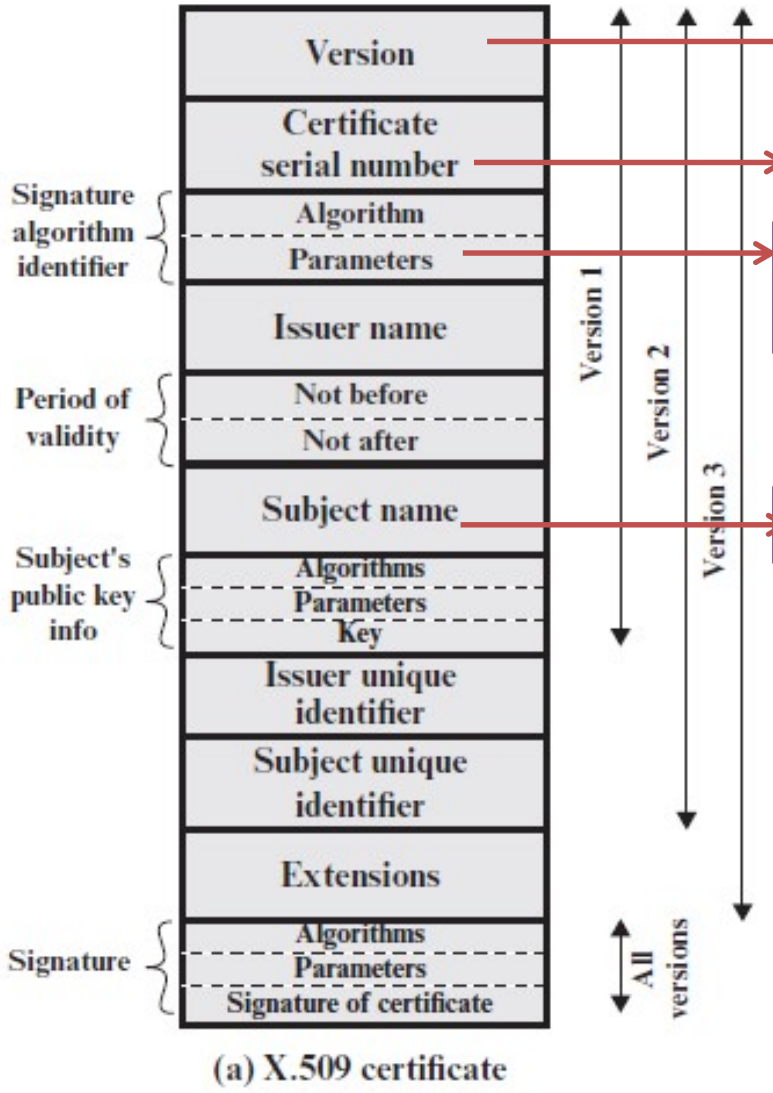
X. 509 Certificates



X. 509 Certificates

1. The heart of the X.509 scheme is the public-key certificate associated with each user.
2. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.
3. The directory server itself is not responsible for the creation of public keys or for the certification function.
4. It merely provides an easily accessible location for users to obtain certificates.

X. 509 Certificates – General Format



(a) X.509 certificate

Differentiates among successive versions of the certificate format; the default is version

An integer value unique within the issuing CA that is unambiguously associated with this certificate.

The algorithm used to sign the certificate together with any associated parameters.

The name of the user to whom this certificate refers.

X. 509 Certificates – General Format

The standard uses the following notation to define a certificate:

$$CA << A >> = CA \{V, SN, AI, CA, UCA, A, UA, A_p, T^A\}$$

where,

Y << X >> = the certificate of user X issued by certification authority Y

CA = name of certificate authority

V = version of the certificate

SN = serial number of the certificate

AI = identifier of the algorithm used to sign the certificate

CA = name of certificate authority

UCA = optional unique identifier of the CA

A = name of user A

UA = optional unique identifier of the user A

Ap = public key of user A

TA = period of validity of the certificate

X. 509 Certificates – Revocation

1. Each certificate includes a period of validity, much like a credit card.
2. Typically, a new certificate is issued just before the expiration of the old one.
3. In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons.
    - The user's private key is assumed to be compromised.
    - The user is no longer certified by this CA. Reasons for this include that the subject's name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA's policies.
    - The CA's certificate is assumed to be compromised.

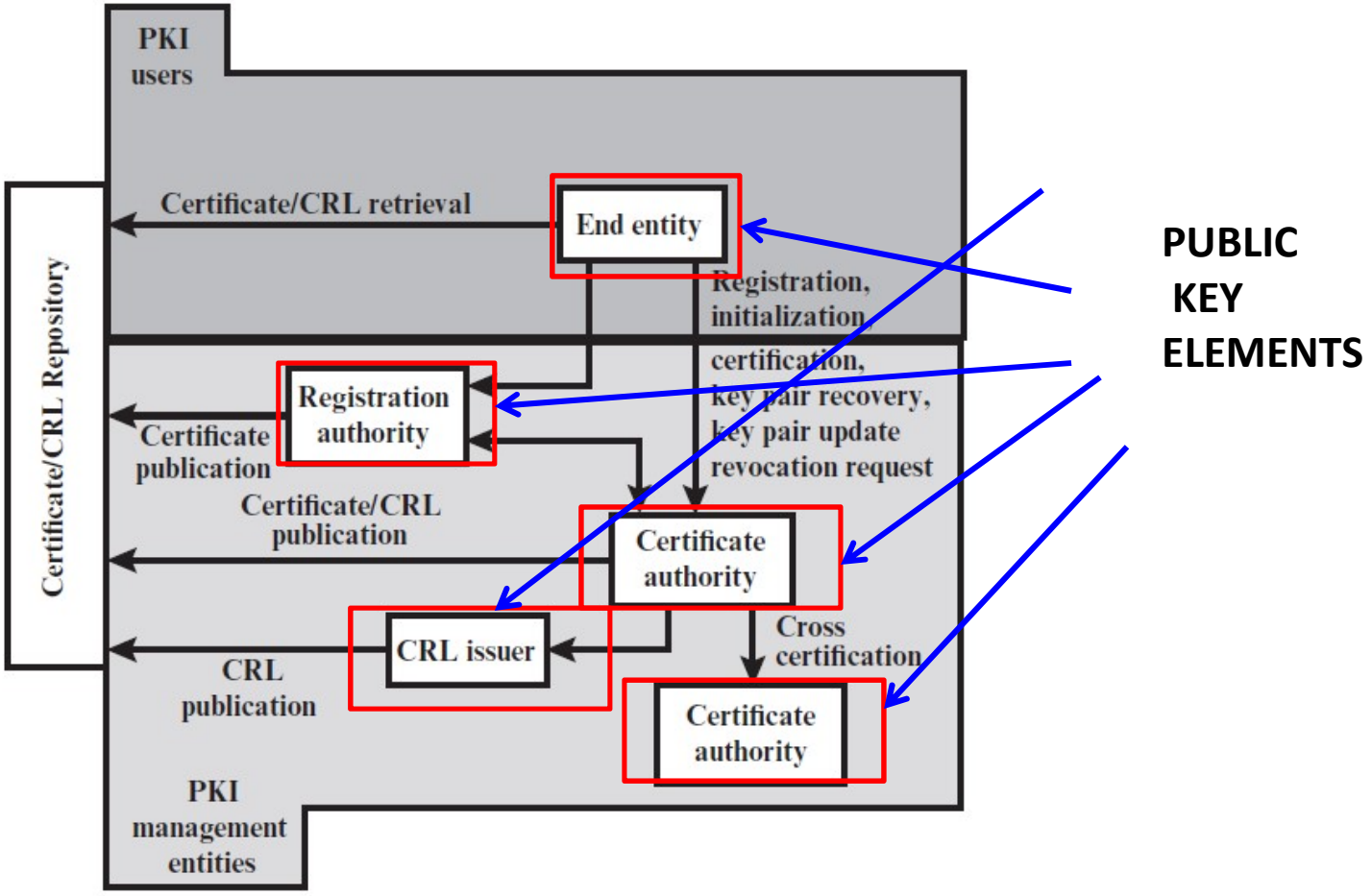
Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. These lists should also be posted on the directory.

PKI – Public Key Infrastructure

1. RFC 4949 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.
2. The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys.

PKI – Public Key Infrastructure

The Formal Model of PKI based on X.509 that is suitable for deploying a certificate-based architecture on the Internet is Public Key Infrastructure X.509 (PKIX)



PKI – Public Key Infrastructure

The key elements of the PKIX model are:

- **End entity:** A generic term used to denote end users, devices. End entities typically consume and/or support PKI related services.
- **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs).
- **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the end entity registration process but can assist in a number of other areas as well.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

PKIX – Management Functions

PKIX identifies a number of management functions that potentially need to be supported by management protocols.

- **Registration:** This is the process whereby a user first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate or certificates for that user.
- **Initialization:** Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure.
- **Certification:** This is the process in which a CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository.

PKIX – Management Functions

- **Key pair recovery:** It is important to provide a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible, otherwise it will not be possible to recover the encrypted data.
- **Key pair update:** All key pairs need to be updated regularly (i.e., replaced with a new key pair) and new certificates issued. Update is required when the certificate lifetime expires and as a result of certificate revocation.
- **Revocation request:** An authorized person advises a CA of an abnormal situation requiring certificate revocation. Reasons for revocation include private key compromise, change in affiliation, and name change.
- **Cross certification:** Two CAs exchange information used in establishing a cross-certificate. A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.