MD5 Algorithm

MD5

- MD5 is quite fast and produces 128 bit message digest
- After some initial processing the input text is processed in 512 bit of blocks.
- This 512 bit blocks are divided into 16, 32 bit sub blocks (16 x 32 = 512)
- The output is set of 4 x 32 bit blocks = 128 bits message digest

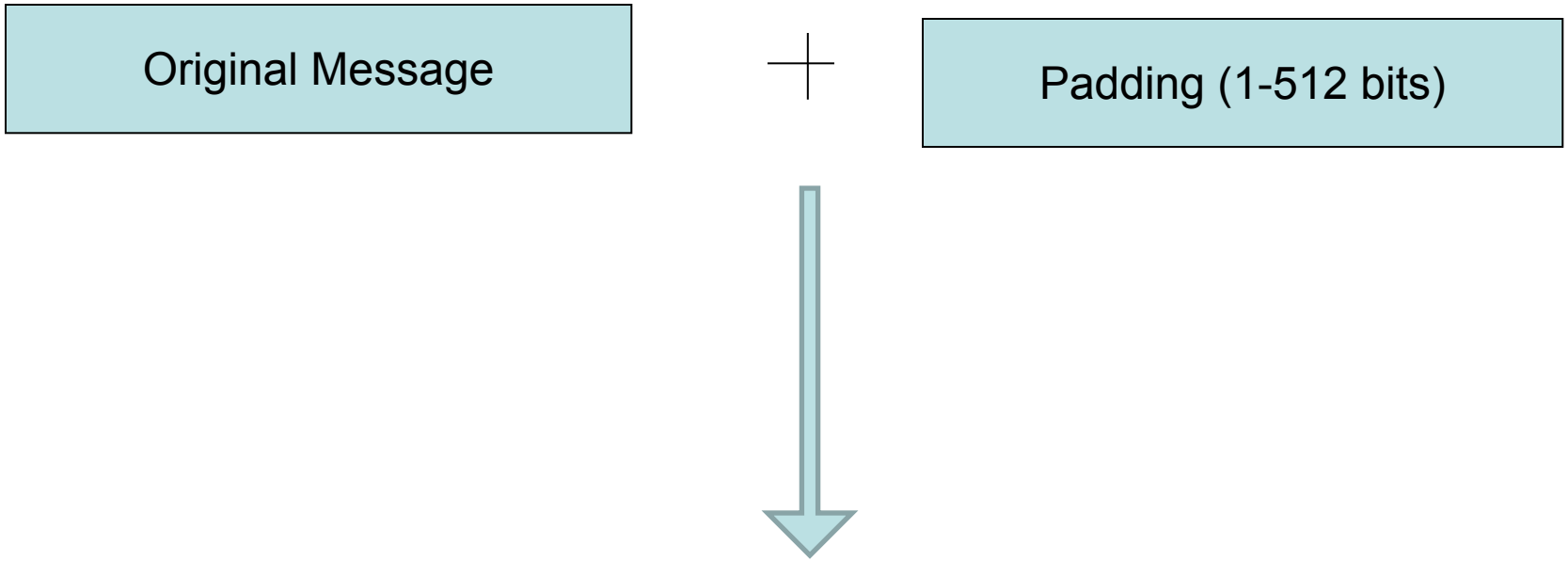
Working of MD5

1. Padding
2. Append length
3. Divide the input into 512 bit blocks
4. Initialize chaining variables
5. Process blocks

Step 1: Padding

- The aim of this step is to make the length of the original message equal to a value which is **64 bit less than the exact multiple of 512**

• 512	512-64	448
• 512 x 2	1024-64	960
• 512 x 3	1536-64	1472



Total length should be 64 bit less than the exact multiple of 512

If 448 bits (448 = 512 – 64)
If 960 bits (960 = 2 x 512 -64)
If 1472 bits (1472 =3 x 512- 64)

Note: Padding is always added even if it is in terms of 512

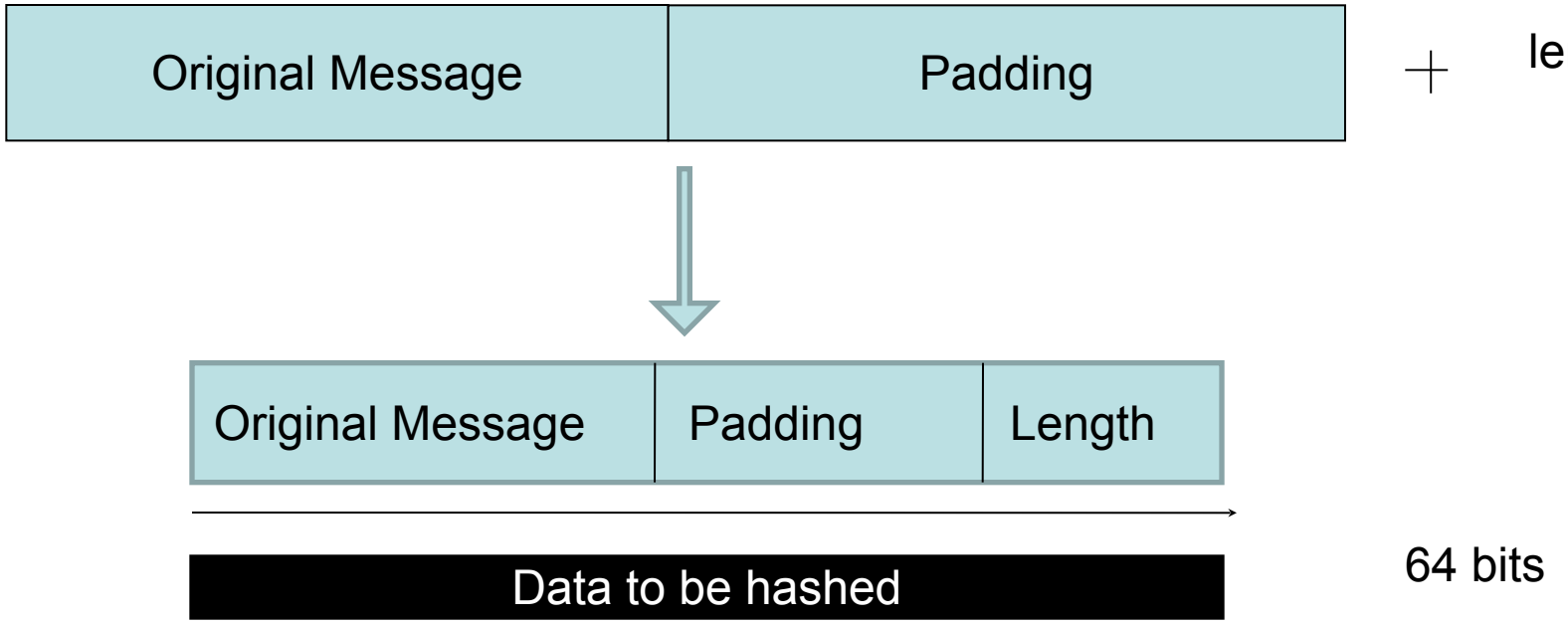
Step 1 cont'd

- If message length is 1000 add 472 = 1472
- If message length is 448 add 512 = 960 , even though 448 is 64 bits less than multiple of 512

Reason		
512	512-64	448
512 x 2	1024-64	960
512 x 3	1536-64	1472

Step 2: Append length

- The length of the original message is Calculated and is appended to the end of the message block + padding

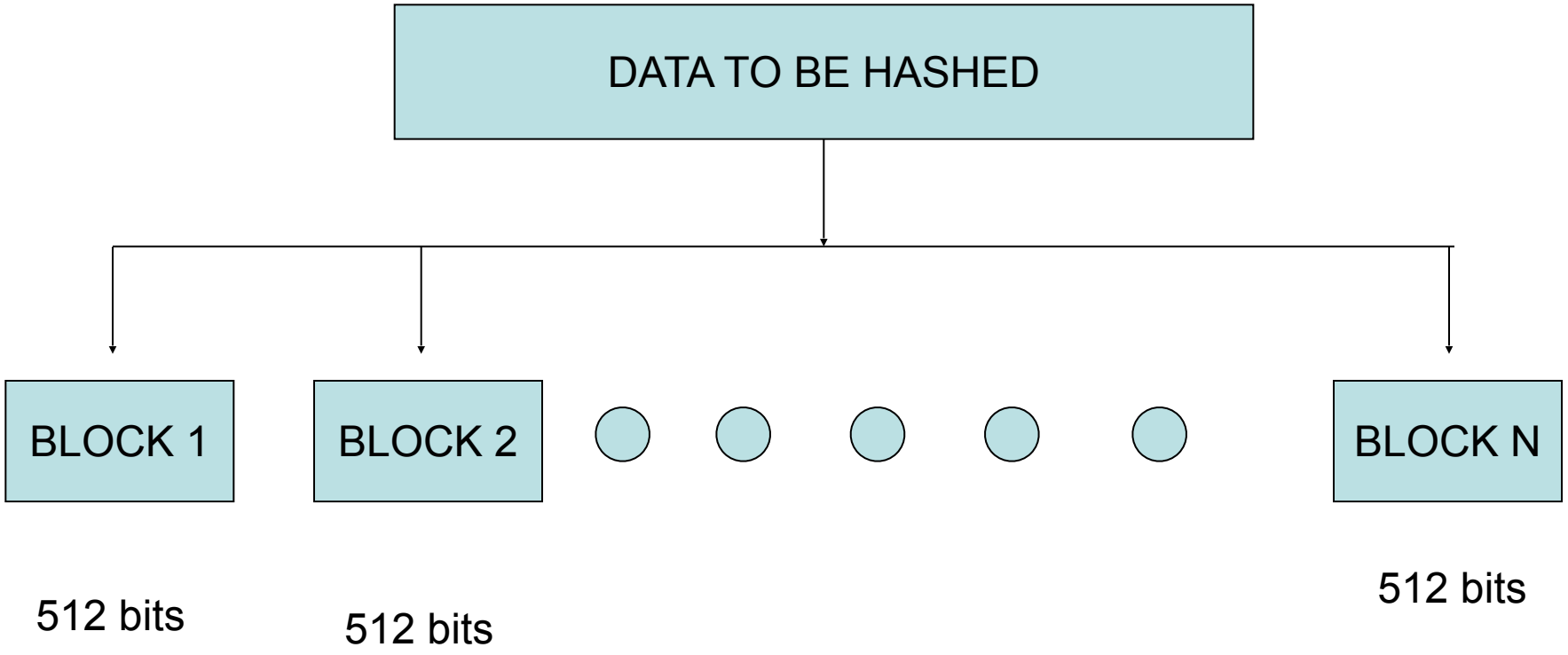


- The length of the original message (excluding the padding) is calculated
- It is appended to the end of the message block + Padding
- The length is expressed in 64 bits
- If the length of the message exceeds 2^{64} bits of length then only last 64 bits are used. (by taking mod 64)
- After 64 bit of length is appended this becomes the final message
- This final message is to be hashed
- The length of the message is in terms of 512 bits

Step 2 cont'd

- If length of original message exceeds 2^{64} then only last 64 bits of the length is used
- After the 64 bits is added this becomes the final message
- The length of the message is now in multiples of 512 bits

Step 3 Divide the i/p into 512 bits blocks



Step 4 – Initialize chaining variables

- Four variables called as chaining variables are initialized
- They are called as A B C and D
- Each of these is a 32 bit no
- The initial hex values are

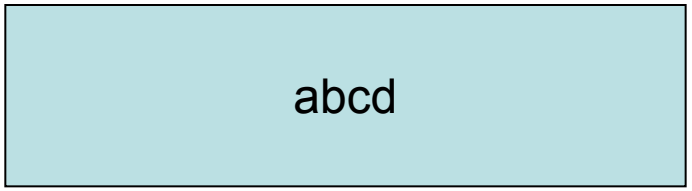
A	**Hex**	**01**	**23**	**45**	**67**
B	**Hex**	**89**	**AB**	**CD**	**EF**
C	**Hex**	**FE**	**DC**	**BA**	**98**
D	**Hex**	**76**	**54**	**32**	**10**

Step 5

- After all the initializations the real algorithm begins
- Its quiet complicated , we will discuss it step by step
- There is a loop that runs for as many 512 bit blocks as are in the message

Step 5.1 : Process blocks

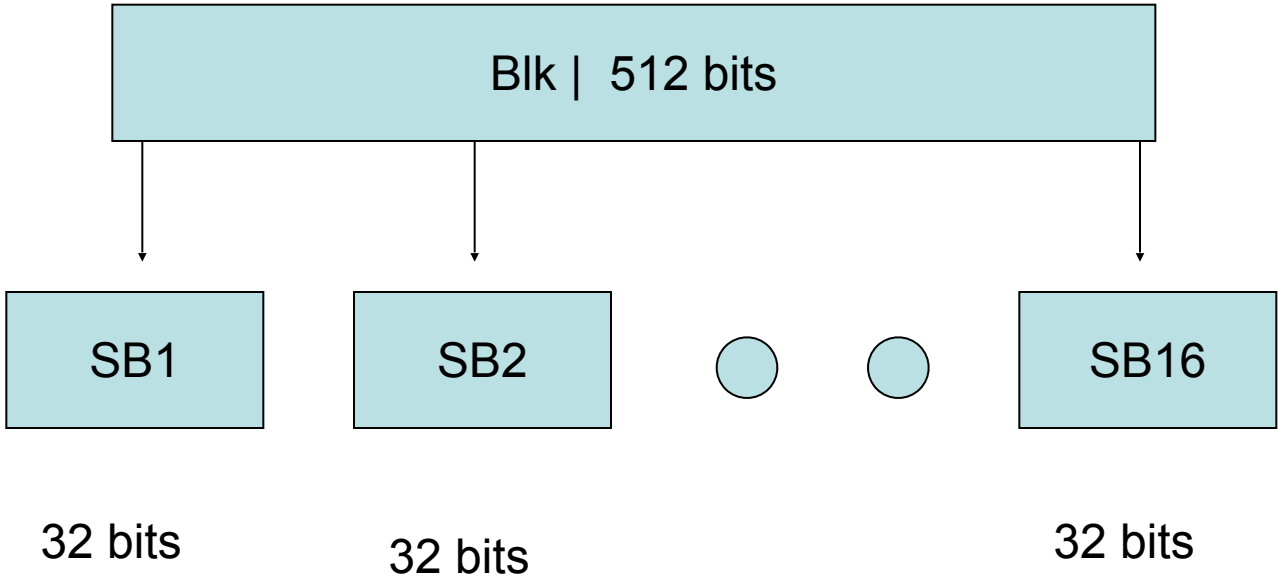
- Copy the 4 chaining variables into 4 corresponding variables a ,b ,c, d
- A -> a
- B -> b
- C -> c
- D -> d



128 bit single register

Step 5.2 Divide

- Divide the current 512 bit block into 16 sub blocks M [i]



Step 5.3

- There are 4 rounds.
- In each round process all the 16 sub blocks
- Inputs to each round are
  - All the 16 sub blocks
  - a, b, c, d
  - Some constants (t)

t

- t is an array of constants
- It contains 64 elements with each element consisting of 32 bits
- t[1] , t[2] ……..t[64]

What is done in these four rounds?

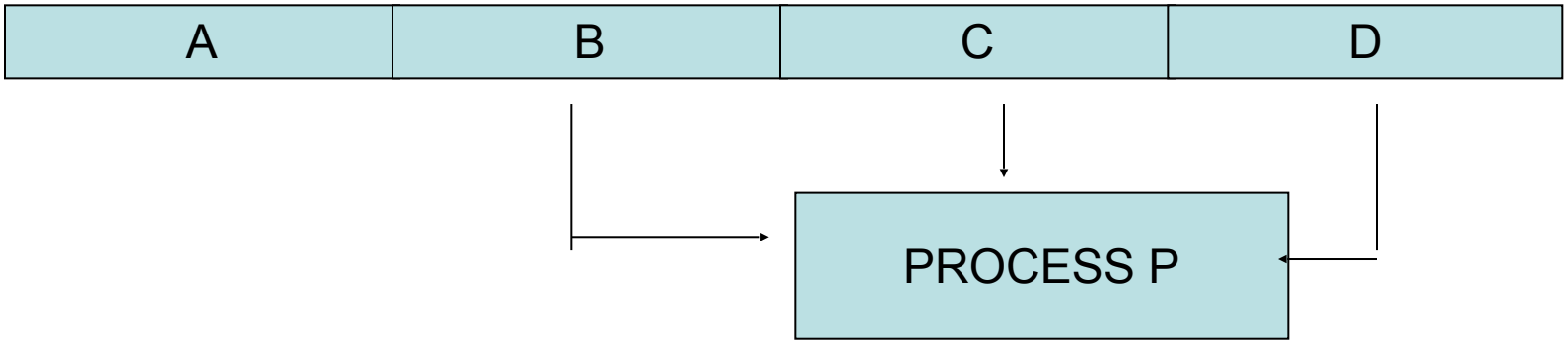
- All the four rounds vary in one major way
- STEP 1 of the rounds has different processing
- The other steps in all the four rounds are the same
  - In each round we have 16 input sub blocks names M[0] , M[1], …. M[15] each of 32 bits
  - We have t[k] , k varies form I to 64,
  - 16 out of 64 values of t is used in each round

Iterations on all 4 rounds

After copying abcd we have 16 such iterations

1. Process P is performed on b,c,d . This process P is different in all the four rounds

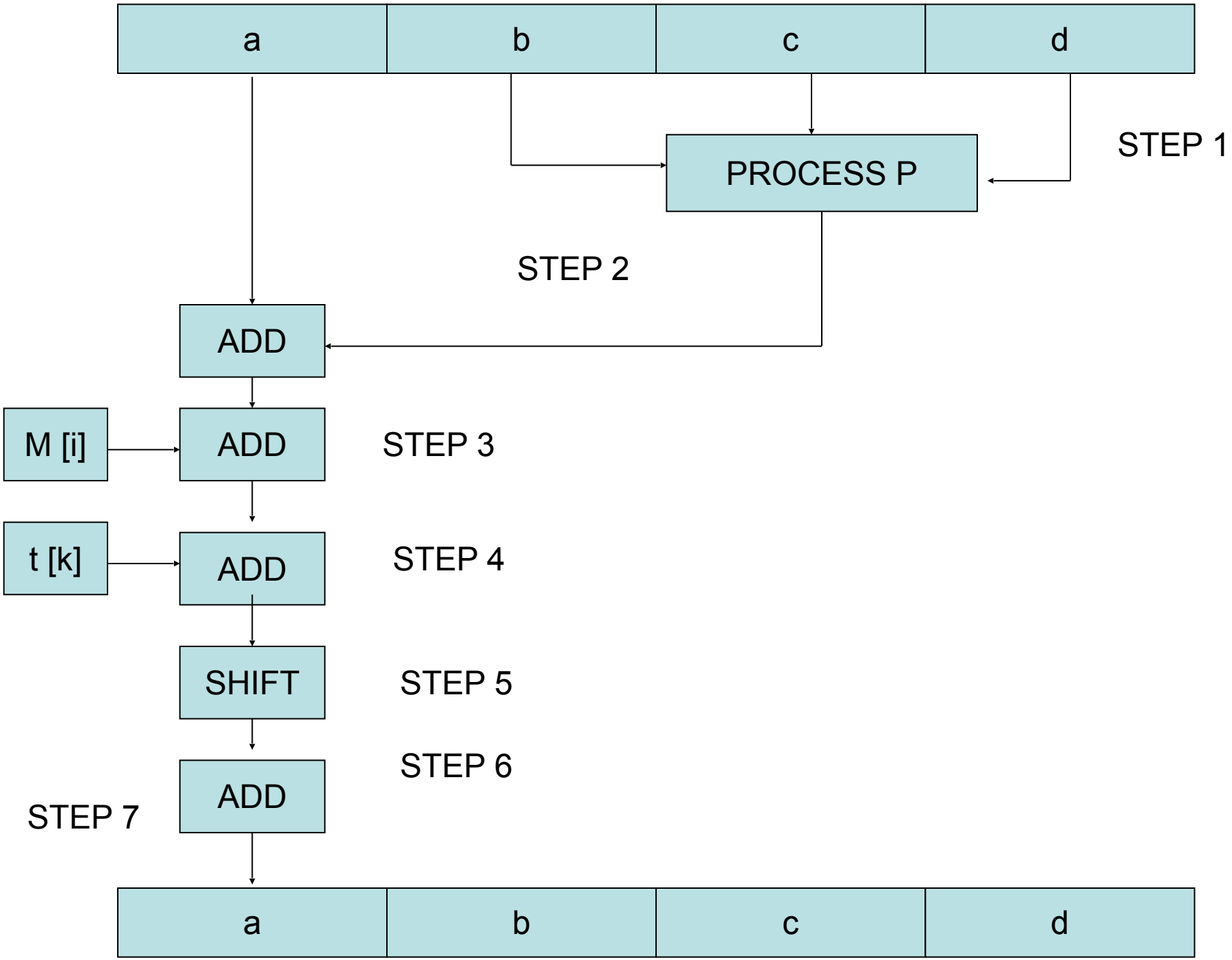
Process P



Process P: some basic Boolean operations on b,c,d

Round	Process P
1	(b AND c) OR ((NOT b) AND (d))
2	(b AND d) OR (c AND (NOT d))
3	(b XOR c XOR d)
4	c XOR (b OR (NOT d))

2. The variable a is added to the output of process P
3. The message sub block M[i] is added to the output of step 2 i.e. to the register abcd
4. The constant t[k] is added to the output of step 3 i.e. to the register abcd
5. The output of step 4 is circular left shifted by s bits
6. The variable b is added to the output of step 5
7. The output of step 6 now becomes abcd for the next step


a
b
c
d
PROCESS P
STEP 1
STEP 2
ADD
M [i]
ADD
STEP 3
t [k]
ADD
STEP 4
SHIFT
STEP 5
STEP 6
ADD
STEP 7
a
b
c
d

Mathematically expressing MD5

- a = b + ((a + process P(b,c,d) + M[i] + T[k] <<< s)
- abcd – chaining variables
- Process P – A non linear operation
- M[i], t[i] – Sub blocks, constants
- <<< s – Circular left shift by s bits

Strength of MD5

- The possibility that 2 messages producing the same message digest using MD5 is in the order of 2^{64}
- Given the message digest working backwards to find the message can lead up to 2^{128} operations

Difference between MD4 and MD5

- Number of rounds
  - Number of rounds in MD4: 3
  - Number of rounds in MD5: 4
- Use of t
  - Same t values in MD4
  - Different t values for all iterations
- Process P
  - Same in MD4
  - Different in MD5

Secure Hash Algorithm (SHA)

- Secure Hash Algorithm (SHA) was developed by NIST along with NSA.
- In 1993, SHA was published as a Federal Information Processing Standard.
- It has following versions-
  - SHA-0
  - SHA-1
  - SHA-2
  - SHA-3

SHA 1

- Padding
- Append Length
- Divide input to 512 bit blocks
- Initialize chaining variables
- Process blocks

Initialize chaining variables

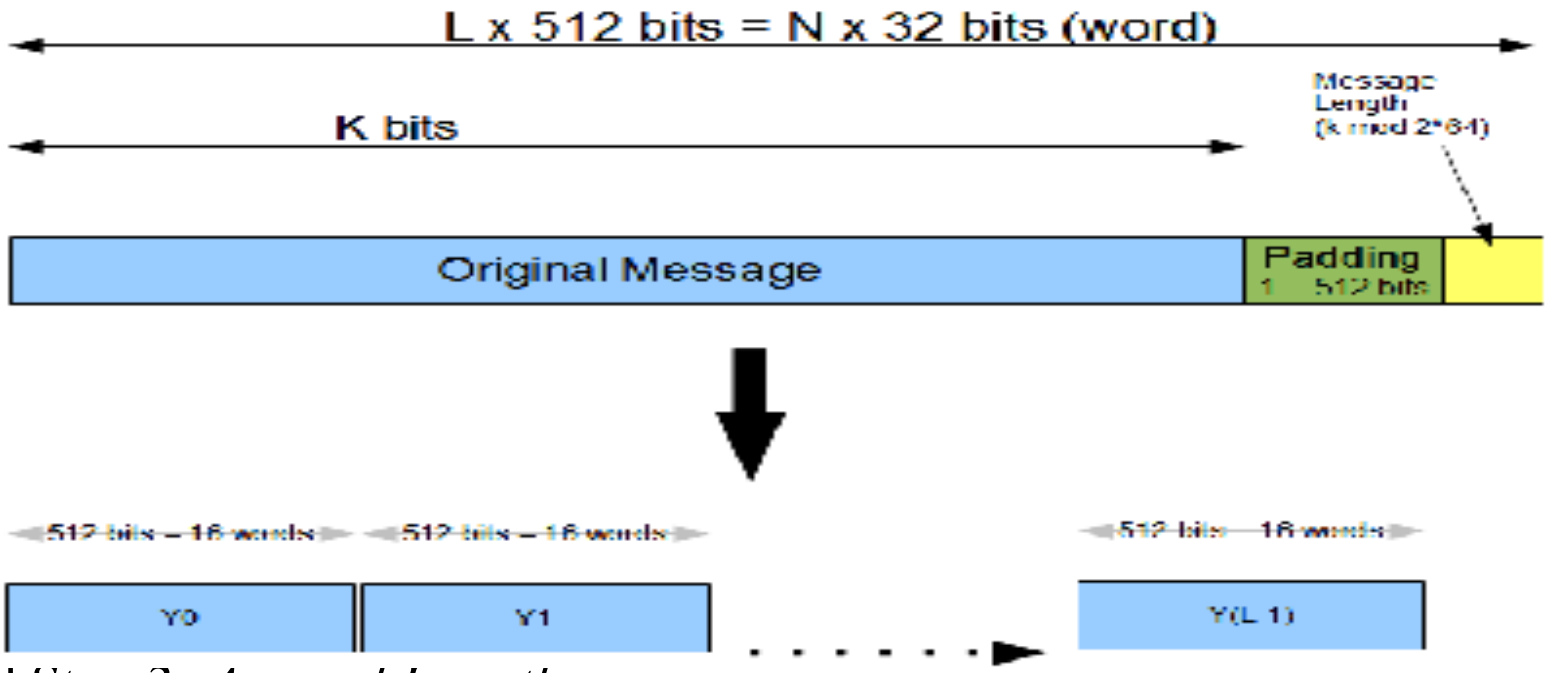
- A- E
- Therefore 5 x 32 = 160 bits as message digest
- Same hex values form A-D
- E – Hex C3 D2 E1 F0

SHA-1

- It works for any input message that is less than 2^{64} bits.
- The output of SHA is a message digest of 160 bits in length.
- This is designed to be computationally infeasible to:
  a) Obtain the original message , given its message digest.
  b) Find two messages producing the same message digest.

How SHA-1 works?

- *Step 1: Padding of Bits*



- *Step 2: Append Length*

- *Step 3: Divide the input into 512-bit blocks*

How SHA-1 works cont…

- *Step 4: Initialize chaining variables*

Chaining Variables	Hex values
A	01 23 45 67
B	89 AB CD EF
C	FE DC BA 98
D	76 54 32 10
E	C3 D2 E1 F0

- *Step 5: Process Blocks-* Now the actual algorithm begins….

How SHA-1 works cont…

- *Step 5.1 :* Copy chaining variables A-E into variables a-e.
- *Step 5.2 :* Divide current 512-bit block into 16 sub-blocks of 32-bits.
- *Step 5.3 :* SHA has 4 rounds, each consisting of 20 steps. Each round takes 3 inputs-
  - » 512-bit block,
  - » The register abcde
  - » A constant K[t] (where t= 0 to 79)

Round	Value of t between
1	1 and 19
2	20 and 39
3	40 and 59
4	60 and 79

How SHA-1 works cont…

- *Step 5.4 :* SHA has a total of 80 iterations (4 rounds X 20 -iterations). Each iteration consists of following operations:-

$$abcde = (e + \text{Process P} + S^5(a) + W[t] + K[t]), a, S^{30}(b) , c , d$$

Where,

abcde = The register made up of 5 variables a, b, c, d, e.

Process P = The logic operation.

S^t = Circular-left shift of 32-bit sub-block by t bits.

W[t[= A 32-bit derived from the current 32-bit sub-block.

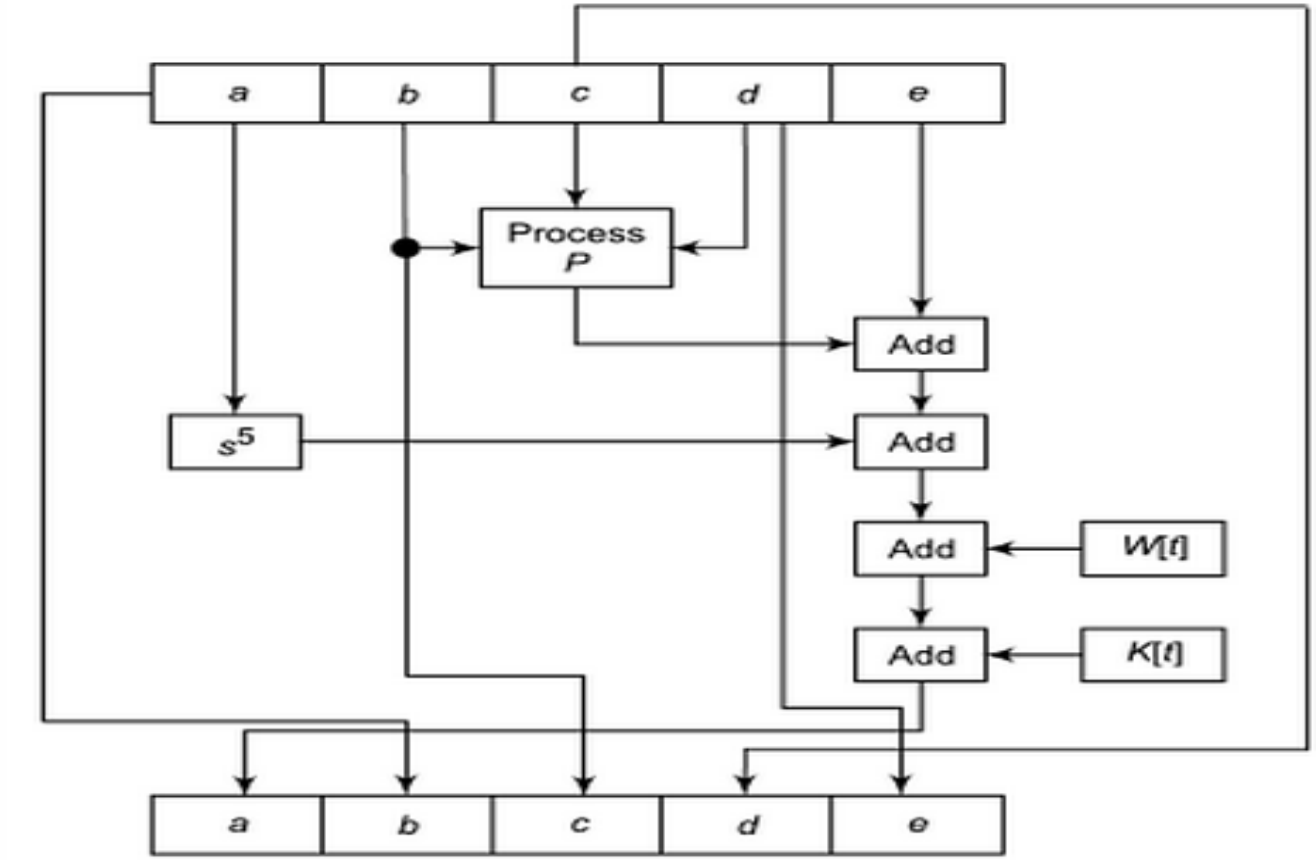
K[t] = One of the five additive constants.

How SHA-1 works cont…

- *Process P in each SHA round*

Round	Process P
1	(b AND c) OR ((NOT b) AND (d))
2	b XOR c XOR d
3	(b AND c) OR (b AND d) OR (c AND d)
4	b XOR c XOR d

How SHA-1 works cont…



Single SHA-1 iteration

How SHA-1 works cont…

- *The values of W[t] are calculated as follows :*
  - For the first 16 words of W (i.e. t=0 to 15) , the contents of the input message sub-block M[t] become the contents of W[t].
  - For the remaining 64 values of W are derived using the equation

$$W[t] = s^1 (W[t-16] \text{ XOR } W[t-14] \text{ XOR } W[t-8] \text{ XOR } W[t-3])$$

Comparison between MD5 and SHA-1

Point of discussion	MD5	SHA-1
Message digest length in bits	128	160
Attack to try and find the original message given a message digest	Requires 2^{128} operations to break in.	Requires 2^{160} operations to break in, therefore more secure.
Attack to try and find two messages producing same message digest	Requires 2^{64} operations to break in.	Requires 2^{80} operations to break in.
Speed	Faster	Slower
Successful attempts so far	There have been reported attempts to some extent.	No such claims so far.

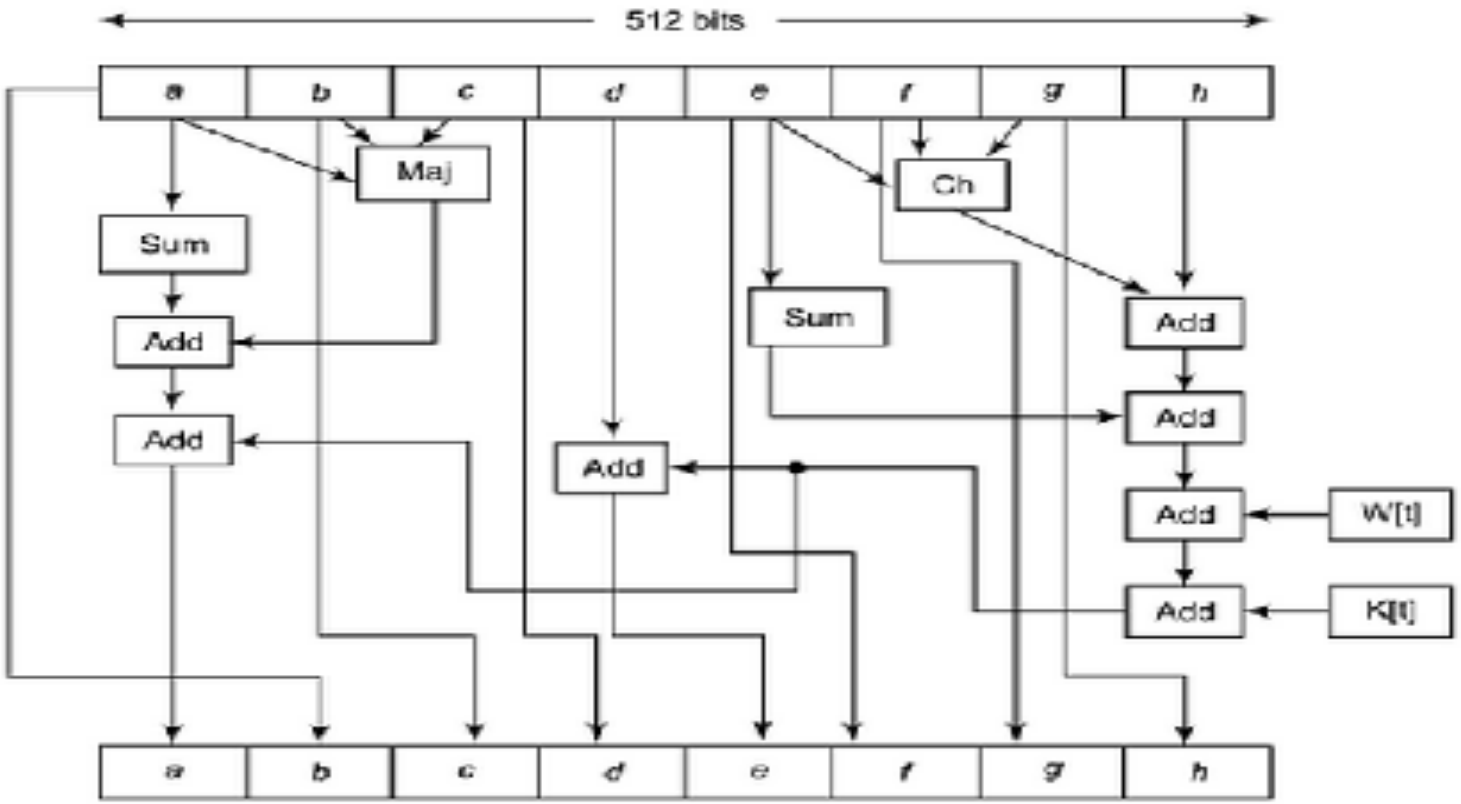
Parameters for various versions of SHA

Parameter	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size(in bits)	160	256	384	512
Message size(in bits)	$<2^{64}$	2^{64}	2^{128}	2^{128}
Block size (in bits)	512	512	1024	1024
Word size (in bits)	32	32	64	64
Steps in algorithm	80	64	80	80

SHA-512

- SHA-512 algorithm takes a message of length 2^{128} bits and produces a message digest of size 512 bits.
- SHA-512 was closely modeled after SHA-1 , which itself is modeled on MD5.

How SHA-512 works cont…



Single SHA-512 iteration

SHA

- Developing Secure Hash Algorithm was initially major concern for defense authorities.
- SHA produces message digest which has an application in digital signature.
- In this way, this technique took a contributed in secure and robust encryption.

HMAC

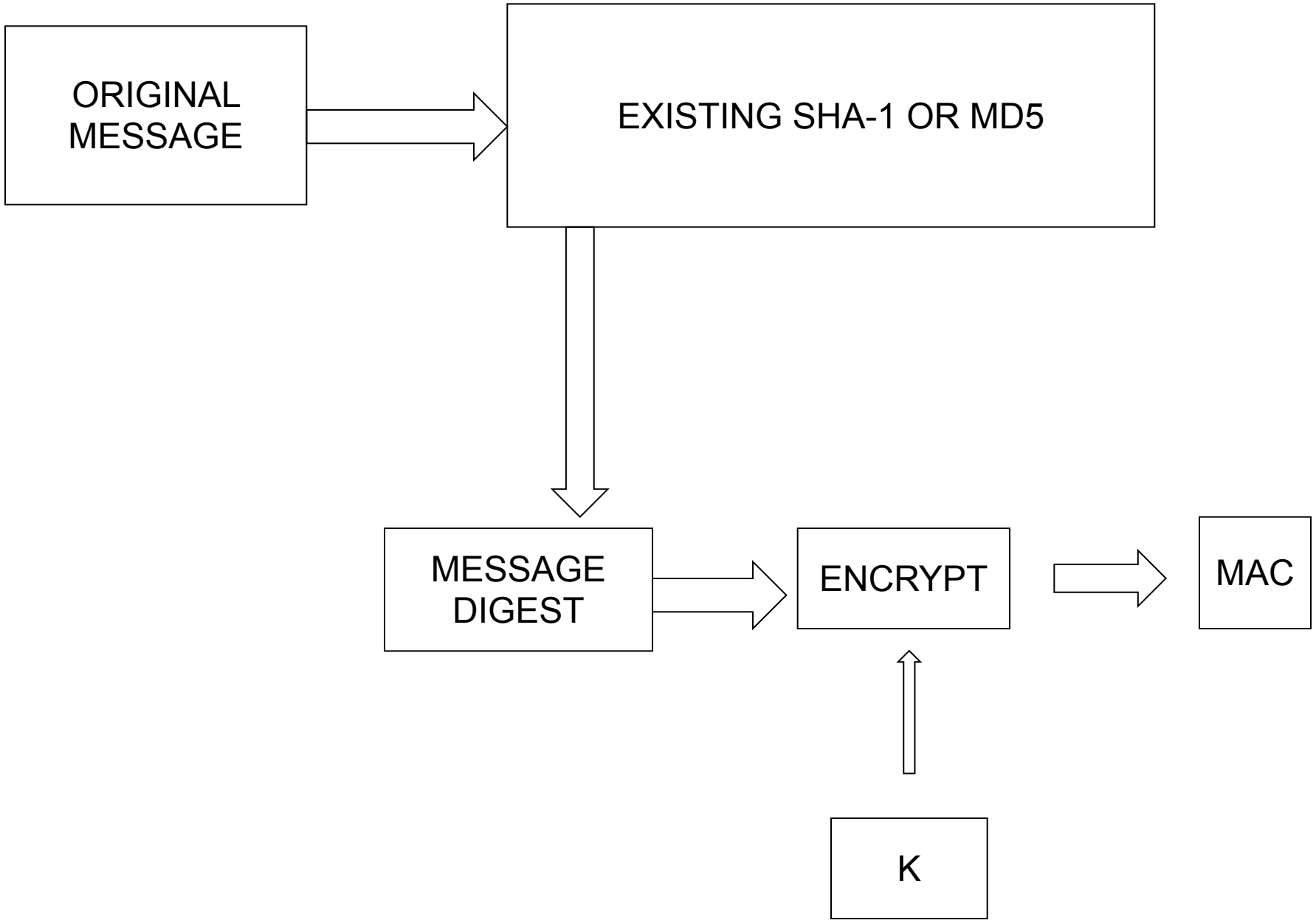
Hash based Message authentication code

Message Authentication

- message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash function

Broader Set of Attacks

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation


ORIGINAL MESSAGE
EXISTING SHA-1 OR MD5
MESSAGE DIGEST
ENCRYPT
MAC
K

MAC Properties

- a MAC is a cryptographic checksum

  $MAC = C_K(M)$

  – C is a function
  – condenses a variable-length message M
  – using a secret key K
- many-to-one function
  – potentially many messages have same MAC
  – but finding these needs to be very difficult

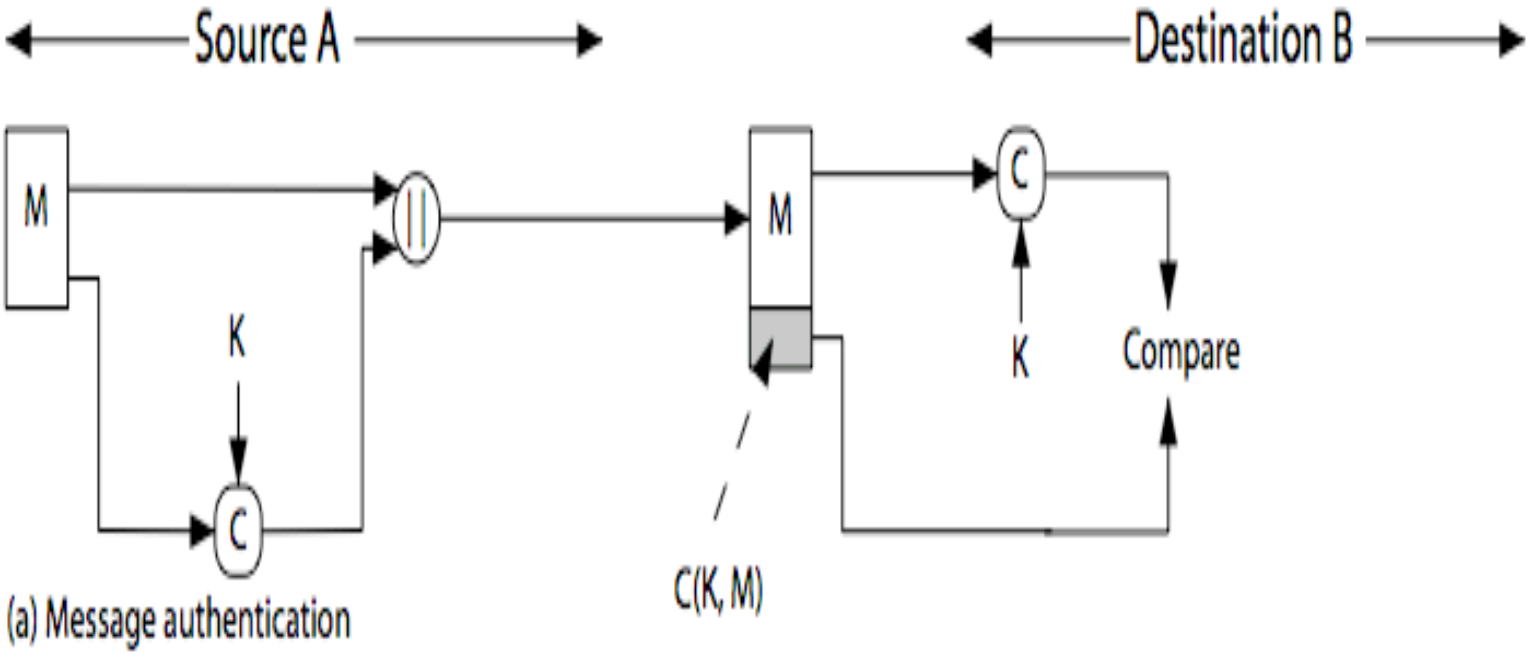
Message Authentication Code (MAC)

- a small fixed-sized block of data:
  - depends on both message and a secret key
  - like encryption though need not be reversible
- appended to message as a **signature**
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender

MAC

- Concept of message authentication code is similar to message digest
- Message digest is only finger print of the message
- In contrast MAC requires that the sender and the receiver should know a shared symmetric secret key which is used in the preparation of MAC
- That is MAC involving cryptographic processing

Message Authentication Code

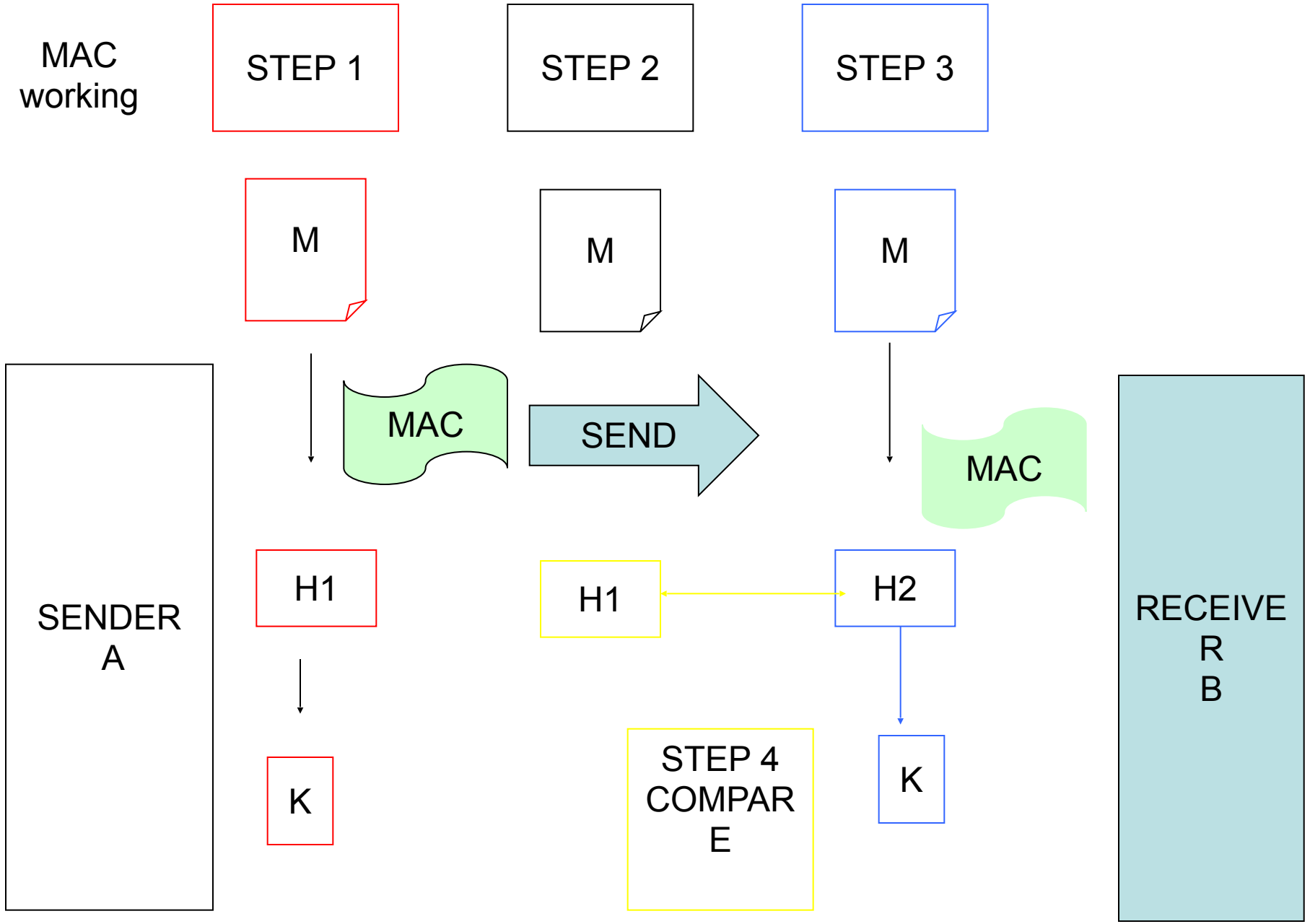


Message Authentication Codes

- MAC provides authentication
- Message can be encrypted for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- why use a MAC?
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (e.g., archival use)
- note that a MAC is not a digital signature

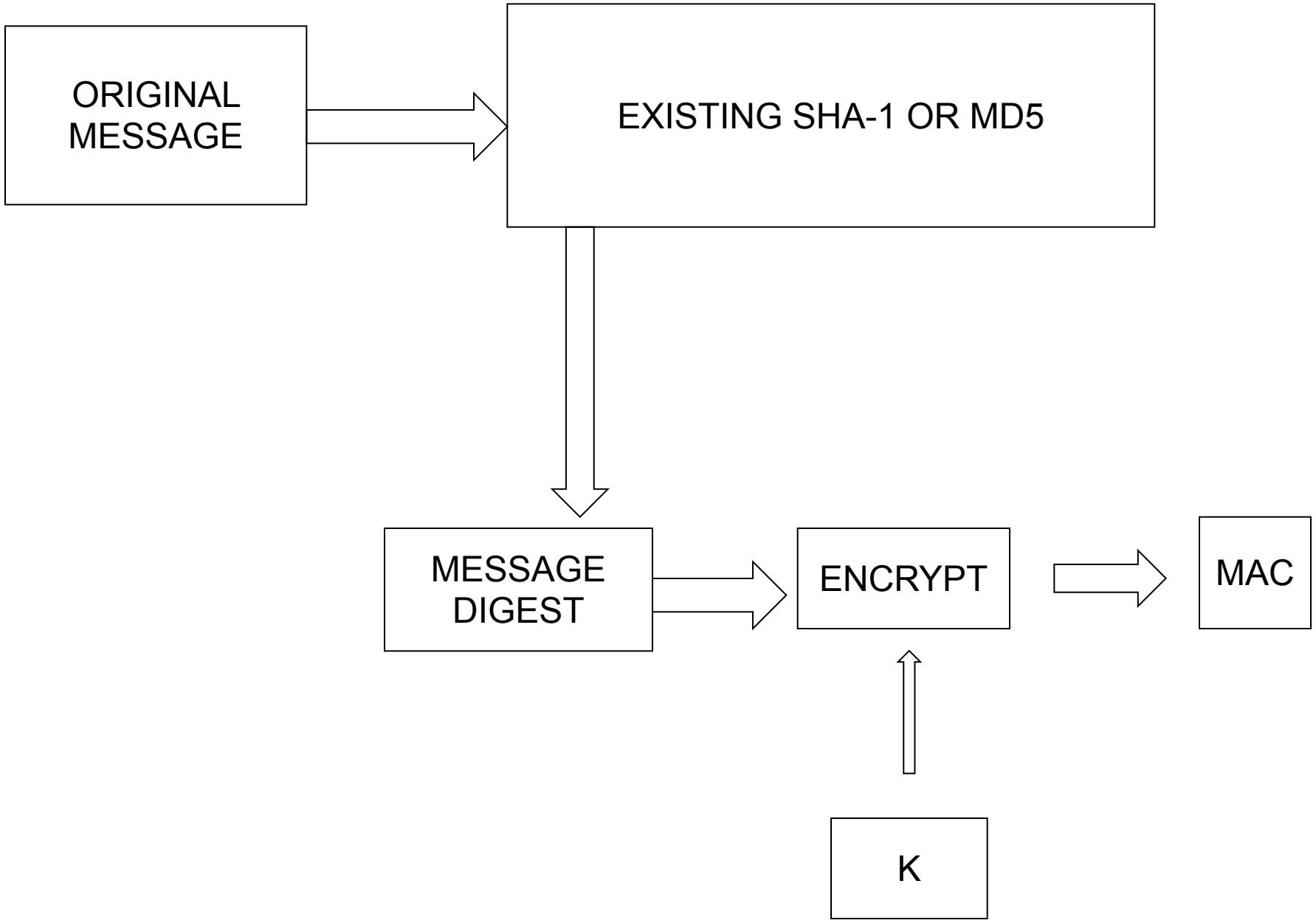
Requirements for MACs

- MAC needs to satisfy the following:
  1. knowing a message and MAC, is infeasible to find another message with same MAC
  2. MACs should be uniformly distributed
  3. MAC should depend equally on all bits of the message


MAC working
STEP 1
STEP 2
STEP 3
M
M
M
SENDER A
MAC
SEND
MAC
RECEIVER B
H1
H1
H2
K
STEP 4 COMPARE
K

HMAC

- It’s a mandatory security implementation in IP security
- Also used in SSL protocol
- Reuses either SHA or MD5 message digest algorithms


ORIGINAL MESSAGE
EXISTING SHA-1 OR MD5
MESSAGE DIGEST
ENCRYPT
MAC
K

INPUTS

- MD – The message digest/hash function used (MD5, SHA-1)
- M – The i/p message whose MAC is to be calculated
- L – The number of blocks in the message M
- b - The number of bits in each block
- K – The shared symmetric key used in HMAC
- ipad - The string 00110110 repeated b/8 times
- opad – The string 01011010 repeated b/8 times

Step 1 (Make K equal to b)

- Generate the key K
- If length (K) < b, expand k to make length equal to number of bits b in original message block (eg K= 170, b=512, add 342 bits all with 0 at the left of K)
- If length (K) = b, no action
- If length (K) > b, trim K to make the length of K equal to B. For this pass through any message digest algorithm, selected for this HMAC instance

Step 2

- XOR K with ipad to produce S1

Step - 3

- Append M to S1

Step - 4

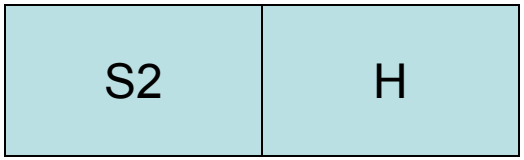
- Select MD5 or SHA-1 to output of step 3
- Output of this operation is H

Step - 5

- XOR K with opad to produce S2

Step - 6

- Append H to S2
- H is the message digest calculated



Step - 7

- Apply MD5 or SHA-1 to output of Step-6
- This is the final MAC

Disadvantages

- As symmetric key is used , exchange of keys is to be carefully carried out
- Even if key is exchanged safely, HMAC cannot be used for more than one receiver
- If more receivers are accepted then how would the receiver know who has sent it? Co receivers can send false message also