Module 3

Approaches to Message Authentication

- A message is authentic when
  - It is not altered(Genuine)
  - It has come from the alleged source
  - It has not be artificially delayed and replayed

Two approaches for Message authentication

- Authentication using conventional Encryption
- Message Authentication without Message Encryption

Message Authentication

- Bob Receives a message from Alice, he wants to know
  - Whether the message was really sent by Alice(Data Origin Authentication)
  - Whether the message has been modified(Data Integrity)
- Solutions
  - Alice attaches a ***MAC*** to message
  - She may either attach ***digital signature*** to the message

Authentication using conventional Encryption

- Symmetric encryption- genuine sender would be able to encrypt a message

- The receiver can recognize a valid message. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper.

- If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.
- Symmetric encryption alone is not a suitable tool for data authentication.
- To give one simple example, in the ECB mode of encryption, if an attacker reorders the blocks of ciphertext, then each block will still decrypt successfully.

Message Authentication without Message Encryption

- An authentication tag is generated and appended to each message for transmission.
- The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.
- Message encryption by itself does not provide a secure form of authentication.
- Message authentication is provided as a separate function from message encryption
- It is possible to combine authentication and confidentiality in a single algorithm by encrypting a message plus its authentication tag.

Message Authentication without Message Encryption

Method 1

- The message must be broadcast in plaintext with an associated message authentication tag.
- The responsible system performs authentication.
- If a violation occurs, the other destination systems are alerted by a general alarm
- This is a cheaper method

Method 2

- One side(generally receiver) has a heavy load and cannot afford the time to decrypt all incoming messages.
- Authentication is carried out on a selective basis with messages being chosen at random for checking.

Message Authentication without Message Encryption

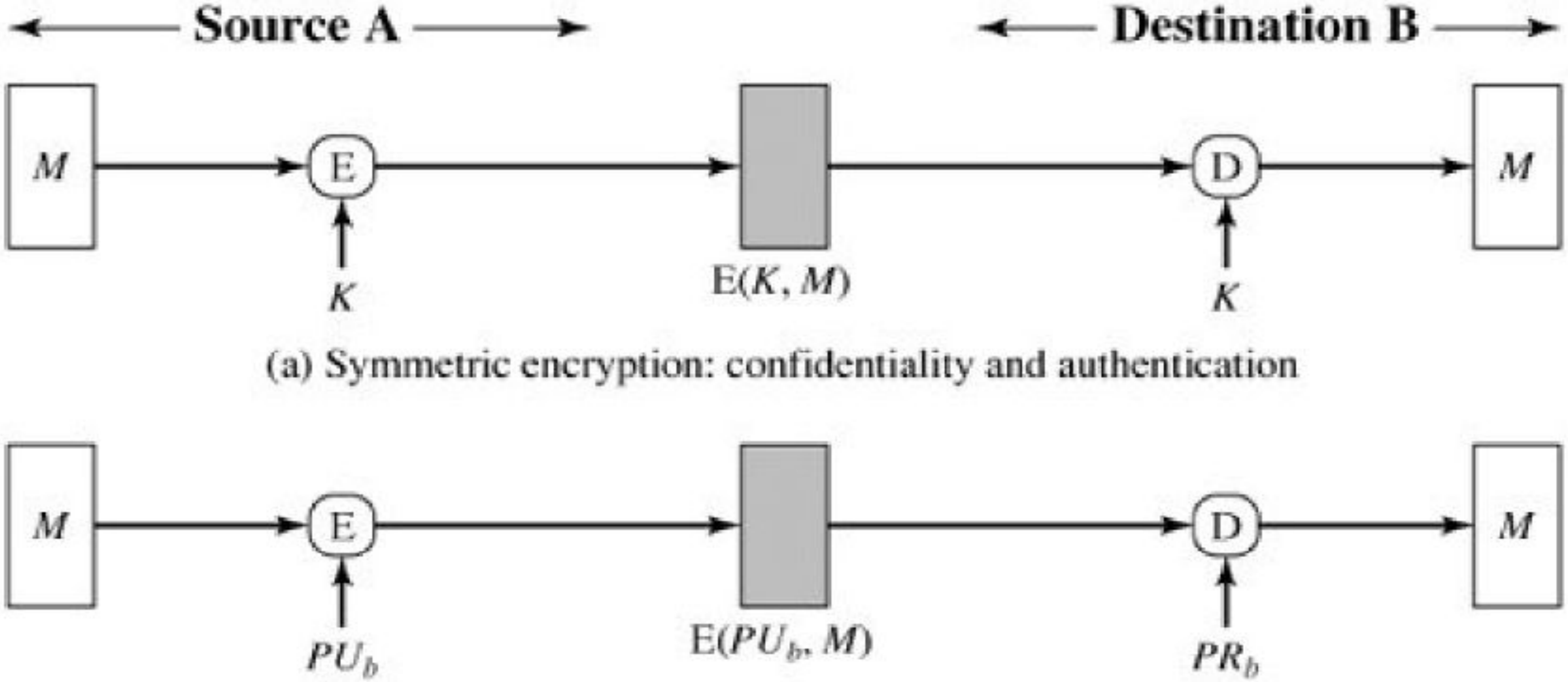
Method 3

- If a message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program

Authentication Functions

- **Message Encryption:** Cipher text of the entire message
- **Message Authentication Code (MAC)**:**A function of the message and a secret key** that produces a fixed-length value that serves as the authenticator
- **Hash Function:** A function that maps a message of any length into a fixed-length hash value, **which serves as the authenticator**

Message Encryption



(a) Symmetric encryption: confidentiality and authentication

(b) Public-key encryption: confidentiality

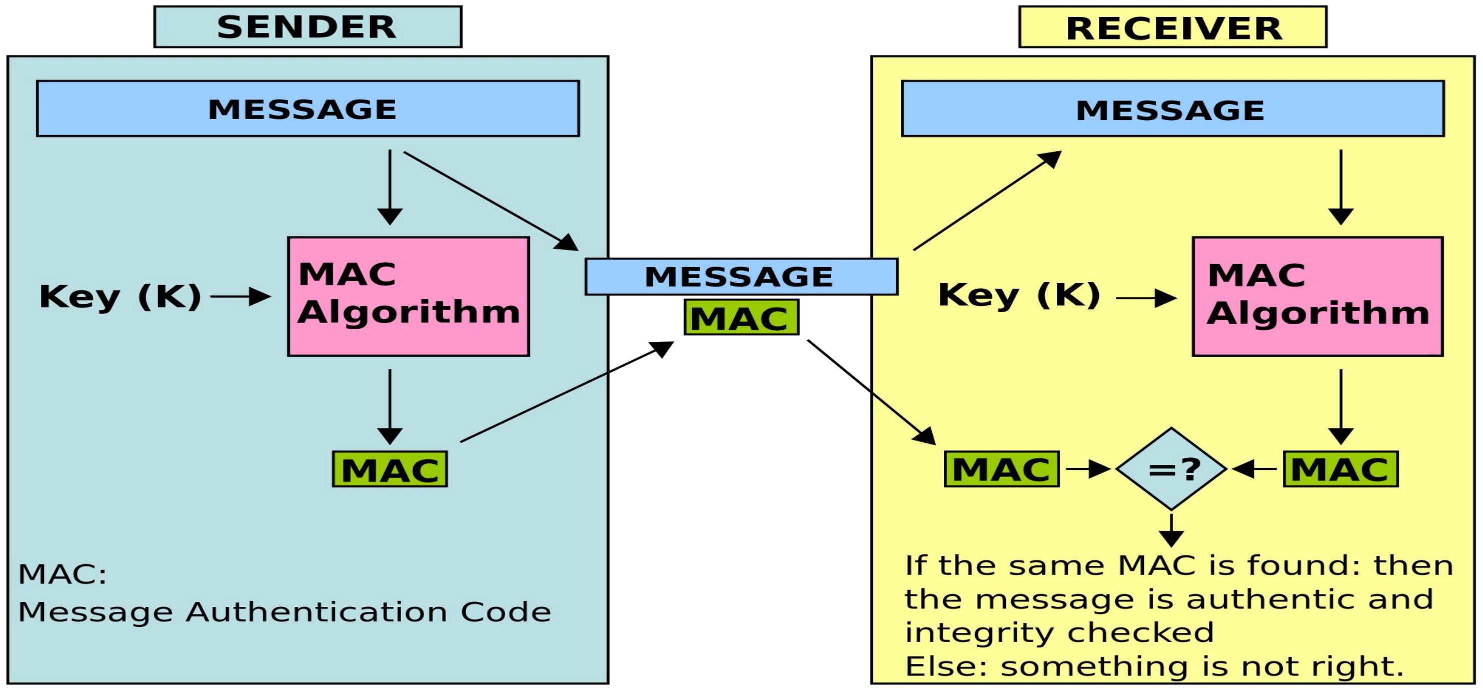
Message Authentication code(MAC)

- One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code (MAC), that is appended to the message.
- This technique assumes that two communicating parties, say A and B, share a common secret key $K_{AB.}$
- When A has a message to send to B, it calculates the message authentication code

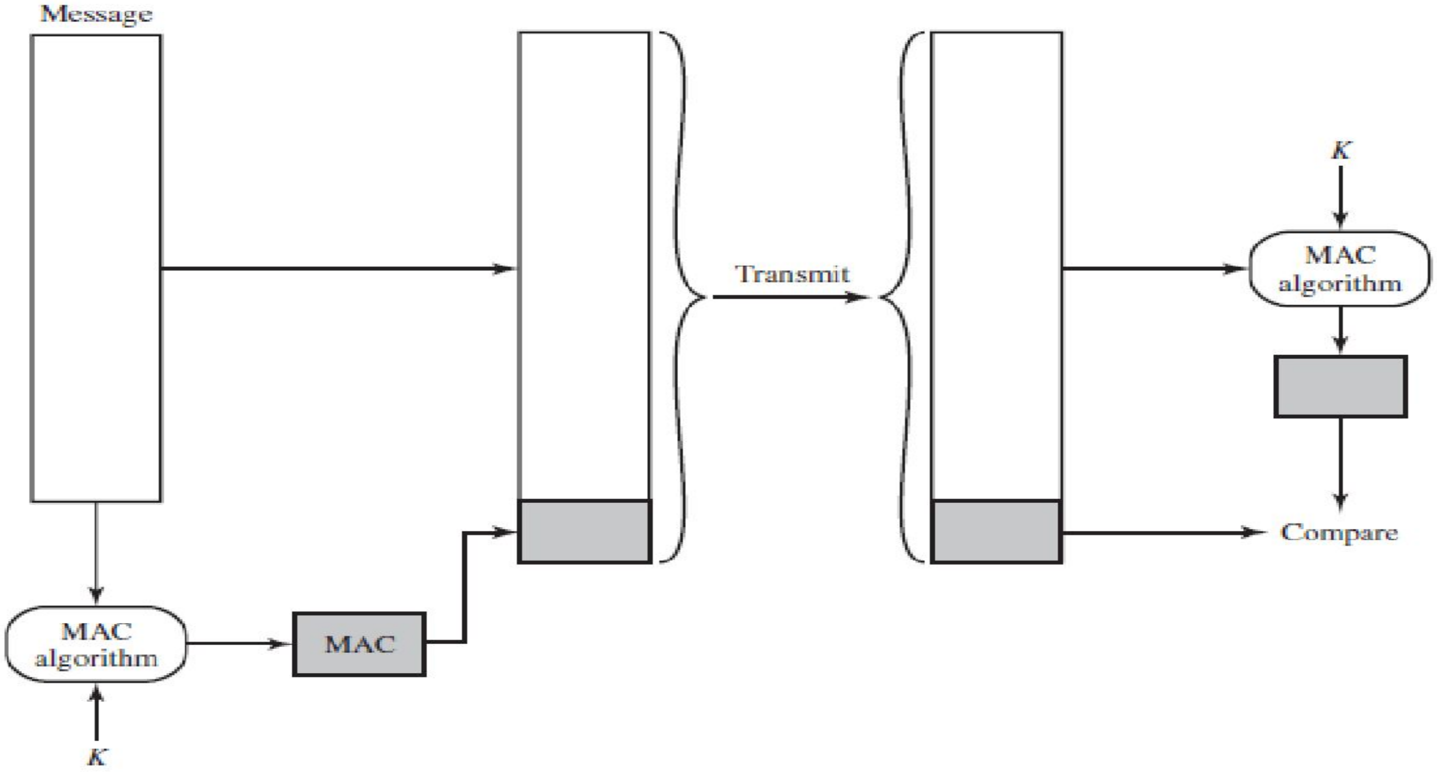
$$MAC_M\ F(K_{AB}, M).T$$

- 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 message authentication code.
- The received code is compared to the calculated code
- The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code.
- If the message includes a sequence number (such as is used with HDLC and TCP), then the receiver can be assured of the proper sequence.

MAC


SENDER
MESSAGE
Key (K)
MAC Algorithm
MAC
MAC:
Message Authentication Code
MESSAGE
MAC
RECEIVER
MESSAGE
Key (K)
MAC Algorithm
MAC
=?
MAC
If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.

MAC


Message
MAC algorithm
K
MAC
Transmit
K
MAC algorithm
Compare

MAC

- The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys
- For example, suppose that we are using **100-bit messages and a 10-bit MAC.**
- Then, there are a total of 2^{100} different messages but only 2^{10} different MACs.
- So, on average, each MAC value is generated by a total of $2^{100}/2^{10} = 2^{90}$ different messages.
- If a 5-bit key is used, then there are $2^{5} = 32$ different mappings from the set of messages to the set of MAC values

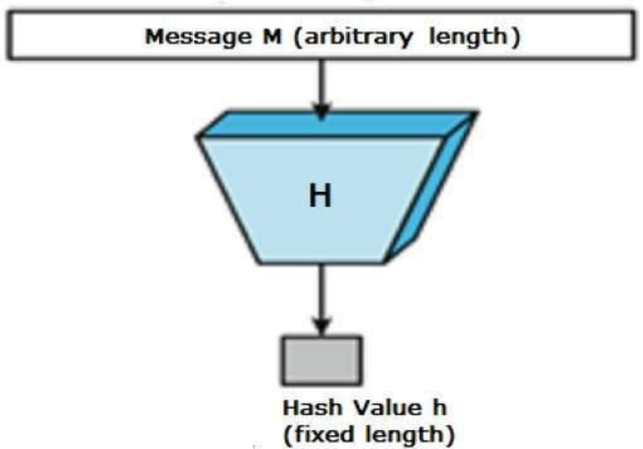
Limitation of MAC

- **Establishment of Shared Secret.**
    - It can provide message authentication among pre-decided legitimate users who have shared key.
    - This requires establishment of shared secret prior to use of MAC.
- **Inability to Provide Non-Repudiation**
    - Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
    - MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.

    - Both these limitations can be overcome by using the public key based digital signatures

Hash Function

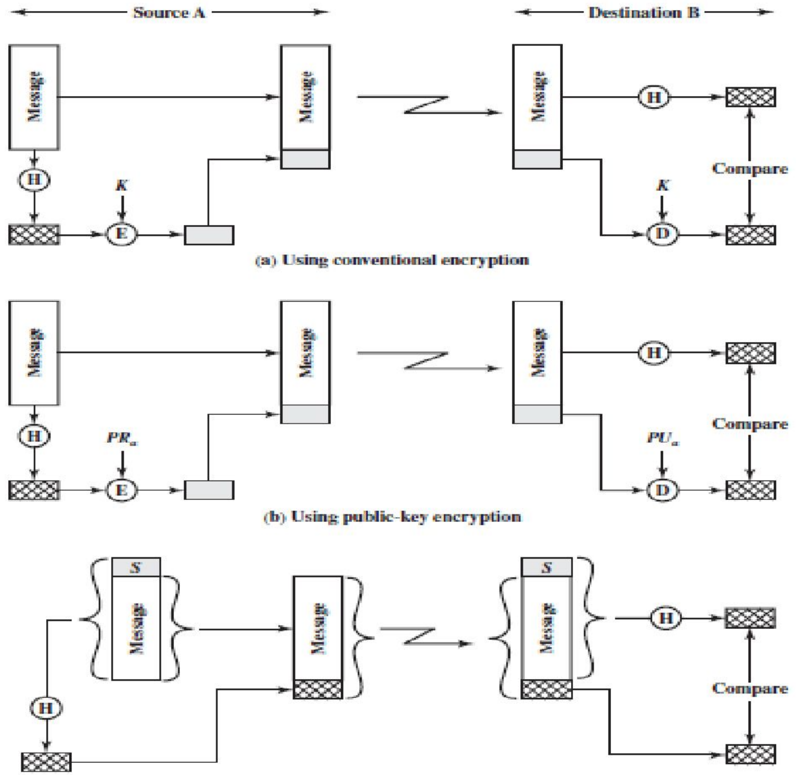
- A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.
- Values returned by a hash function are called **message digest** or simply **hash values**.



One Way Hash Functions

- Like message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size message digest H(M) as output.
- Unlike the MAC, a hash function does not take a 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.

One way Hash Functions

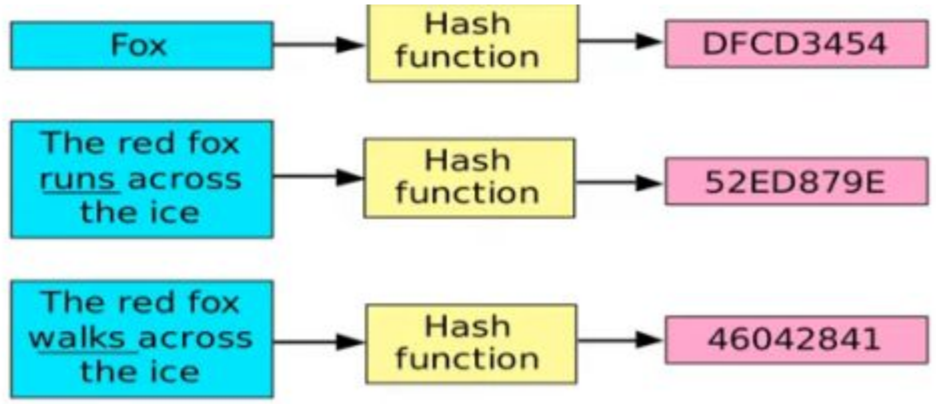


(a) Using conventional encryption

(b) Using public-key encryption

(c) Using secret value

Hash function



Popular Hash Functions

- MD5
  - was most popular and widely used hash function for quite some years.
  - The MD family comprises of hash functions MD2, MD4, MD5 and MD6..It is a 128-bit hash function.
- Secure Hash Function (SHA)
  - Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3.
  - The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.
  - SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.
  - SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending up on number of bits in their hash value.

Popular Hash Functions

- RIPEMD
  - The RIPEMD is an acronym for RACE Integrity Primitives Evaluation Message Digest. This set of hash functions was designed by open research community and generally known as a family of European hash functions.
  - Original RIPEMD (128 bit) is based upon the design principles used in MD4
- Whirlpool
  - This is a 512-bit hash function.
  - Three versions of Whirlpool have been released; namely WHIRLPOOL-0, WHIRLPOOL-T, and WHIRLPOOL.

Features of Hash Functions

- **Fixed Length Output (Hash Value)**
  - Hash function coverts data of arbitrary length to a fixed length. This process is often referred to as **hashing the data**.
  - The hash is much smaller than the input data, hence hash functions are sometimes called **compression functions**.
  - Since a hash is a smaller representation of a larger data, it is also referred to as a **digest**.
  - Popular hash functions generate values between 160 and 512 bits.
- **Efficiency of Operation**
  - Generally for any hash function h with input x, computation of h(x) is a fast operation.
  - Computationally hash functions are much faster than a symmetric encryption.

Properties of Hash Functions

- **Pre-Image Resistance**
  - This property means that it should be computationally hard to reverse a hash function.
  - If a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
  - This property protects against an attacker who only has a hash value and is trying to find the input.
- **Second Pre-Image Resistance**
  - This property means given an input and its hash, it should be hard to find a different input with the same hash.
  - If a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
  - This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.
  - It is also known as **weak collision resistant**

Properties of Hash Functions

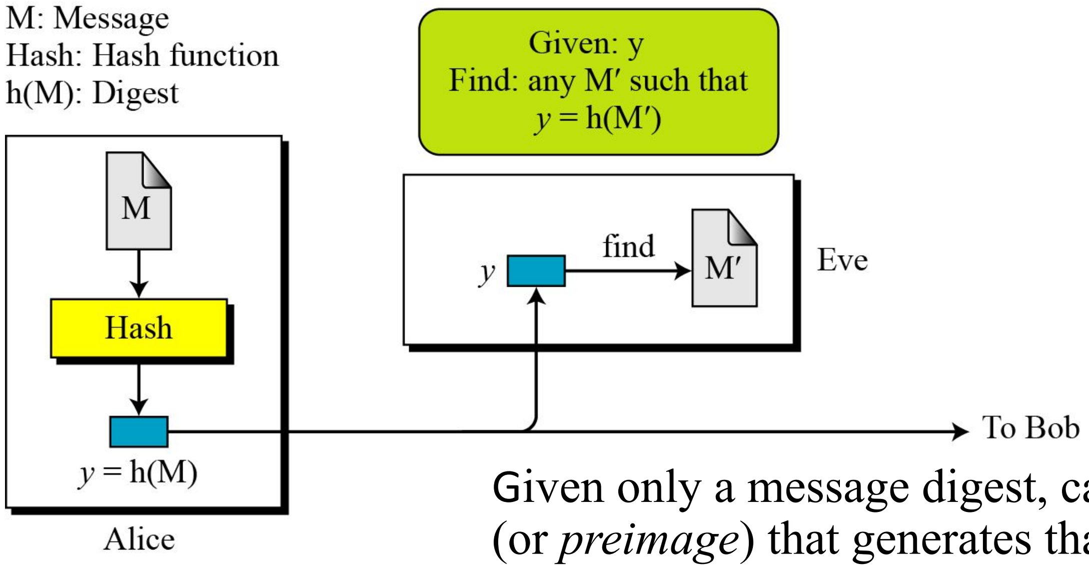
- **Collision Resistance**
  - This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
  - For a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
  - Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
  - This property makes it very difficult for an attacker to find two input values with the same hash.
  - If a hash function is collision-resistant **then it is second pre-image resistant.**

Preimage resistant

- This measures how difficult to devise a message which hashes to the known digest

Preimage Attack

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



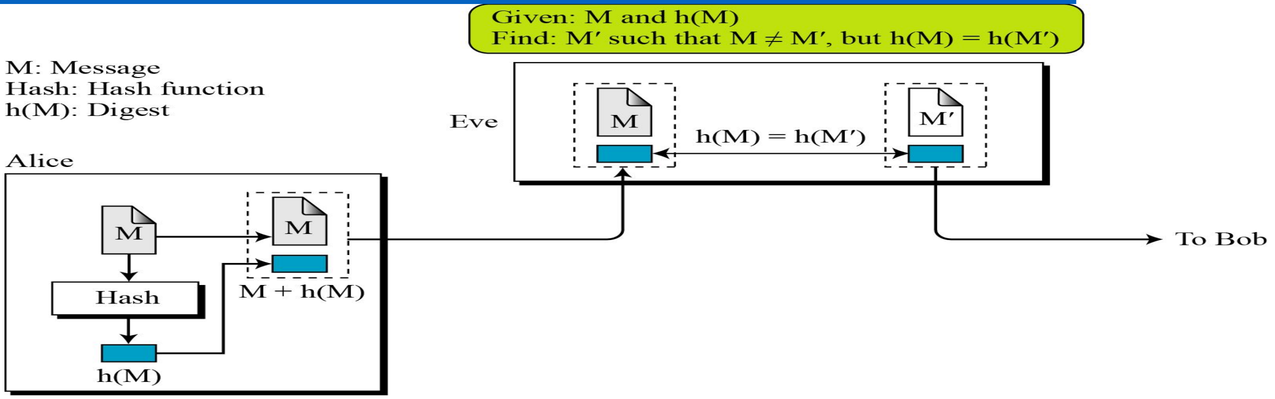
Given only a message digest, can't find any message (or *preimage*) that generates that digest.

Second preimage resistant

- This measures how difficult to devise a message which hashes to the known digest and its message

Second Preimage Attack

Given: M and h(M) **Find: M′ ≠ M such that h(M) = h(M′)**



- Given one message, can't find another message that has the same message digest. An attack that finds a second message with the same message digest is a *second pre-image* attack.
  - It would be easy to forge new digital signatures from old signatures if the hash function used weren't second preimage resistant

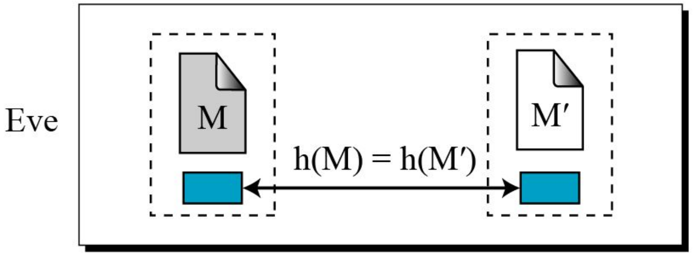
Collision Resistant

Collision Attack

Given: none **Find: M′ ≠ M such that h(M) = h(M′)**

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

Find: M and M′ such that M ≠ 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

Hash function property-Summary

Deterministic

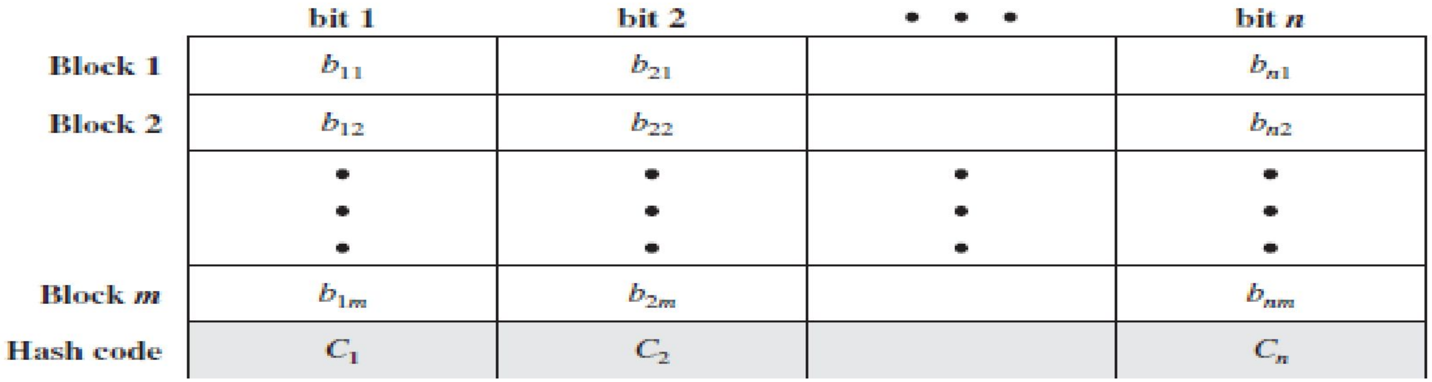
Fast computation

Pre Image Resistance

Avalanche Effect

Collision resistance

Simple hash function



	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

Simple hash function

- All hash functions operate using the following general principles.
- The input (message, file, etc.) is viewed as a sequence of *n*-bit blocks.
- The input is processed one block at a time in an iterative fashion to produce an *n*-bit hash function.
- The two simple techniques are given.

Simple Hash Function

One of the simplest hash function is the bit by bit X-OR operation.

- $C_i = b_{i1} \quad b_{i2} \quad \ldots\ldots\ b_{im}$
- Ci = *i*th bit of the hash code, $1 \leq i \leq n$
- M = number of n-bit blocks of the input
- b *ij* = ith bit in jth block

Two way Hash function

- A simple way to improve is to perform a one bit circular shift , or rotation, on the hash value after each block is processed.
- **The procedure can be summarized as follows:**

1. Initially set the n-bit hash value to zero
2. **Process each successive n-bit block of data as follows**

- Rotate the current hash value to the left by each bit.
- X-OR the block into the Hash value

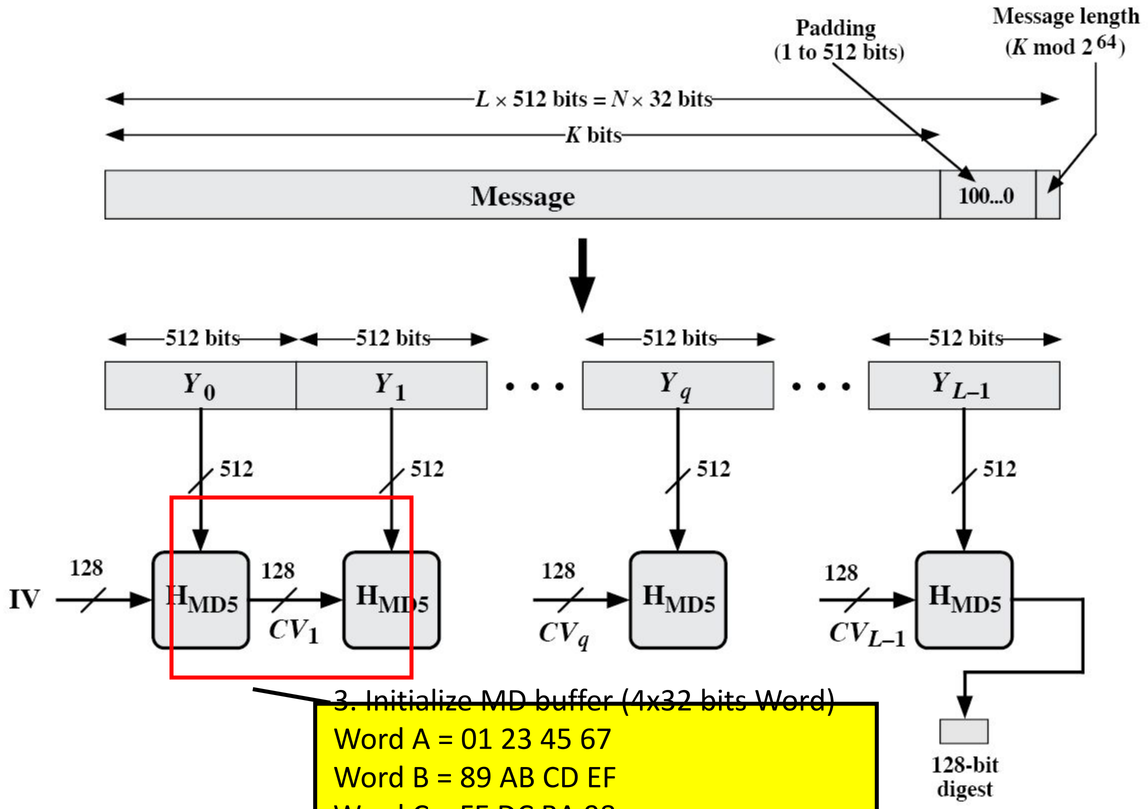
Simple hash function

- bit-by-bit exclusive-OR (XOR) of every block
    - $C_i = b_{i1}\ xor\ b_{i2}\ xor \ldots xor\ b_{im}$
    - a longitudinal redundancy check
    - reasonably effective as data integrity check
- one-bit circular shift on hash value
    - for each successive *n-bit* block
        - rotate current hash value to left by1bit and XOR block
    - good for data integrity but useless for security

- The second procedure provides a good measure of data integrity, it is virtually useless for data security when an encrypted hash code is used with a plaintext message.

- Given a message, it is an easy matter to produce a new message that yields that hash code: Simply prepare the desired alternate message and then append an *n*-bit block that forces the combined new message plus block to yield the desired hash code.

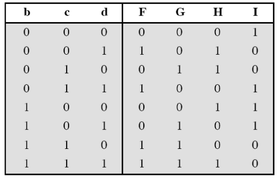
MD5 Overview



2. Append length (64bits)

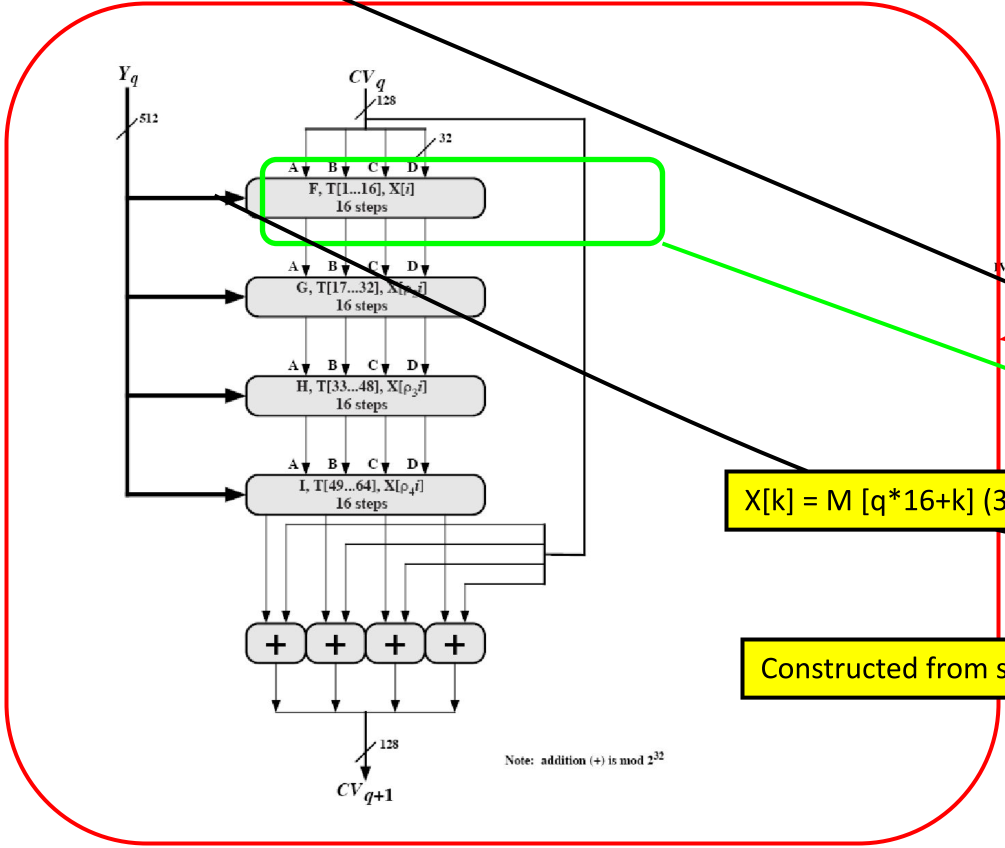
1. Append padding bits (to 448 mod 512)

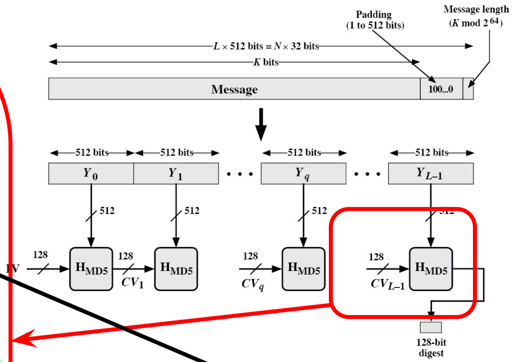
3. Initialize MD buffer (4x32 bits Word)
Word A = 01 23 45 67
Word B = 89 AB CD EF
Word C = FE DC BA 98
Word D = 76 54 32 10

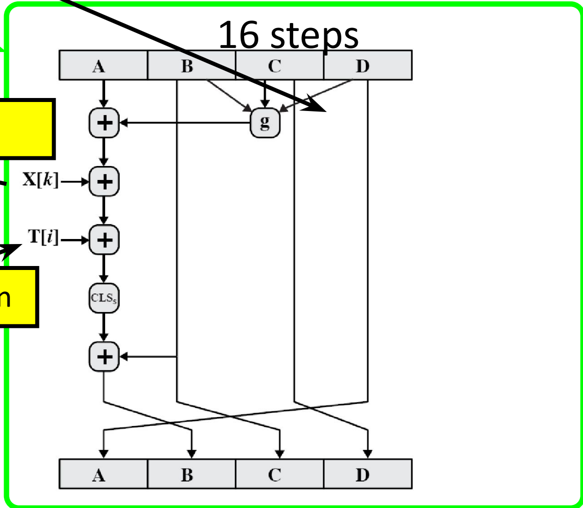


b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

Hash Algorithm Design – MD5


Y_q
512
CV_q
128
32
A B C D
F, T[1...16], X[i]
16 steps
G, T[17...32], X[$\rho_2 i$]
16 steps
H, T[33...48], X[$\rho_3 i$]
16 steps
I, T[49...64], X[$\rho_4 i$]
16 steps
128
CV_{q+1}
Note: addition (+) is mod 2^{32}





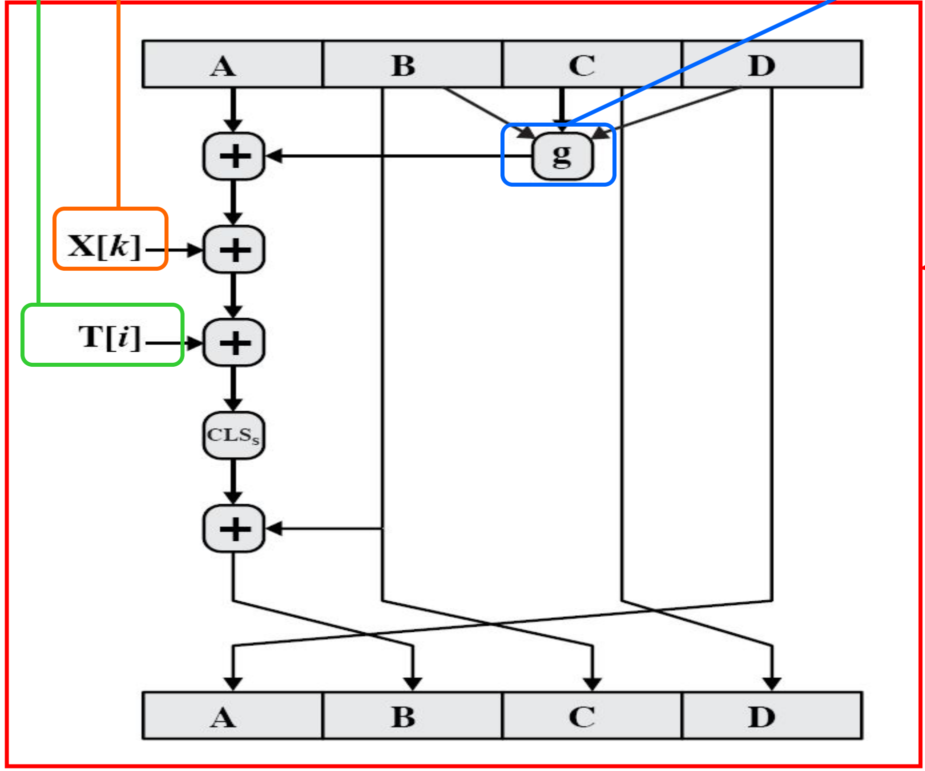
X[k] = M [q*16+k] (32 bit)

Constructed from sine function

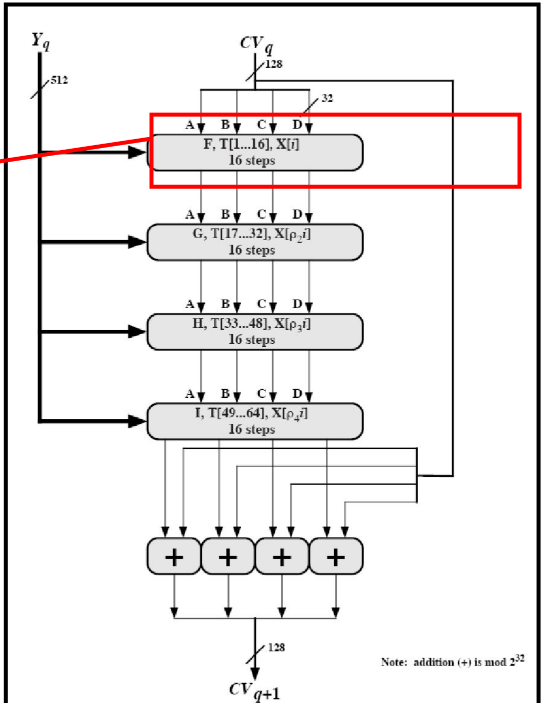
The ith 32-bit word in matrix T, constructed from the sine function

M [q*16+k] = the kth 32-bit word from the qth 512-bit block of the msg

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$



Single step

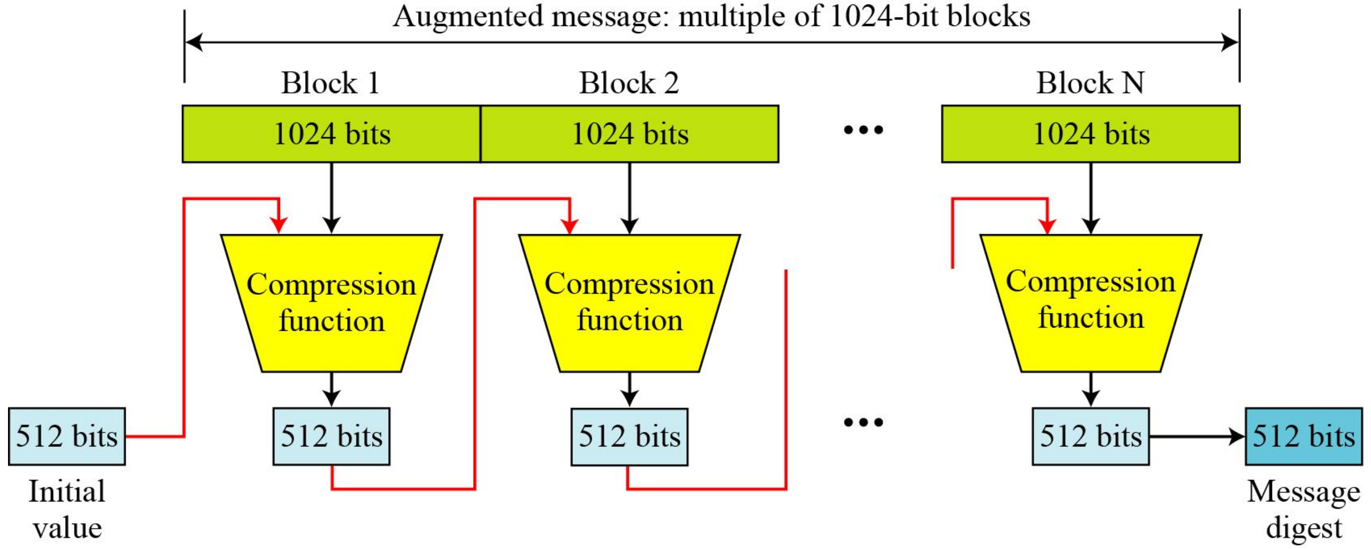


Secure Hash functions

Requirements of secure hash functions are

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(*x*) is relatively easy to compute for any given *x*, making both hardware and software implementations practical.
4. For any given code *h*, it is computationally infeasible to find *x* such that H(*x*) = *h.* A hash function with this property is referred to as **one-way** or **preimage resistant**.3
5. For any given block *x*, it is computationally infeasible to find *y* ≠ *x* with H(*y*)= H(*x*). A hash function with this property is referred to as **second preimage resistant**. This is sometimes referred to as **weak collision resistant**.
6. It is computationally infeasible to find any pair (*x*, *y*) such that H(*x*) H(*y*).A hash function with this property is referred to as **collision resistant**. This is sometimes referred to as **strong collision resistant**. It is resistant against birthday attacks

SHA-512 Overview



Secure Hash Algorithms (SHAs)

- **(SHA) was developed by** the National Institute of Standards and Technology (NIST) and published as a **federal information processing standard** (FIPS 180) **in 1993**.
- *A* revised version was issued **in 1995** and is generally referred to as SHA-1.
- **SHA is based on the hash function MD4** and its design closely models MD4
- Several Hash Algorithms (SHA) were **designed by Ron Rivest.**
- These are referred to as MD2, MD4 and MD5 where MD stands for **message digest.**
- **MD5 is the strengthened version of MD4 and** uses messages of blocks of 512 bits and creates a 128 bits digest.

Secure Hash Algorithms (SHAs)

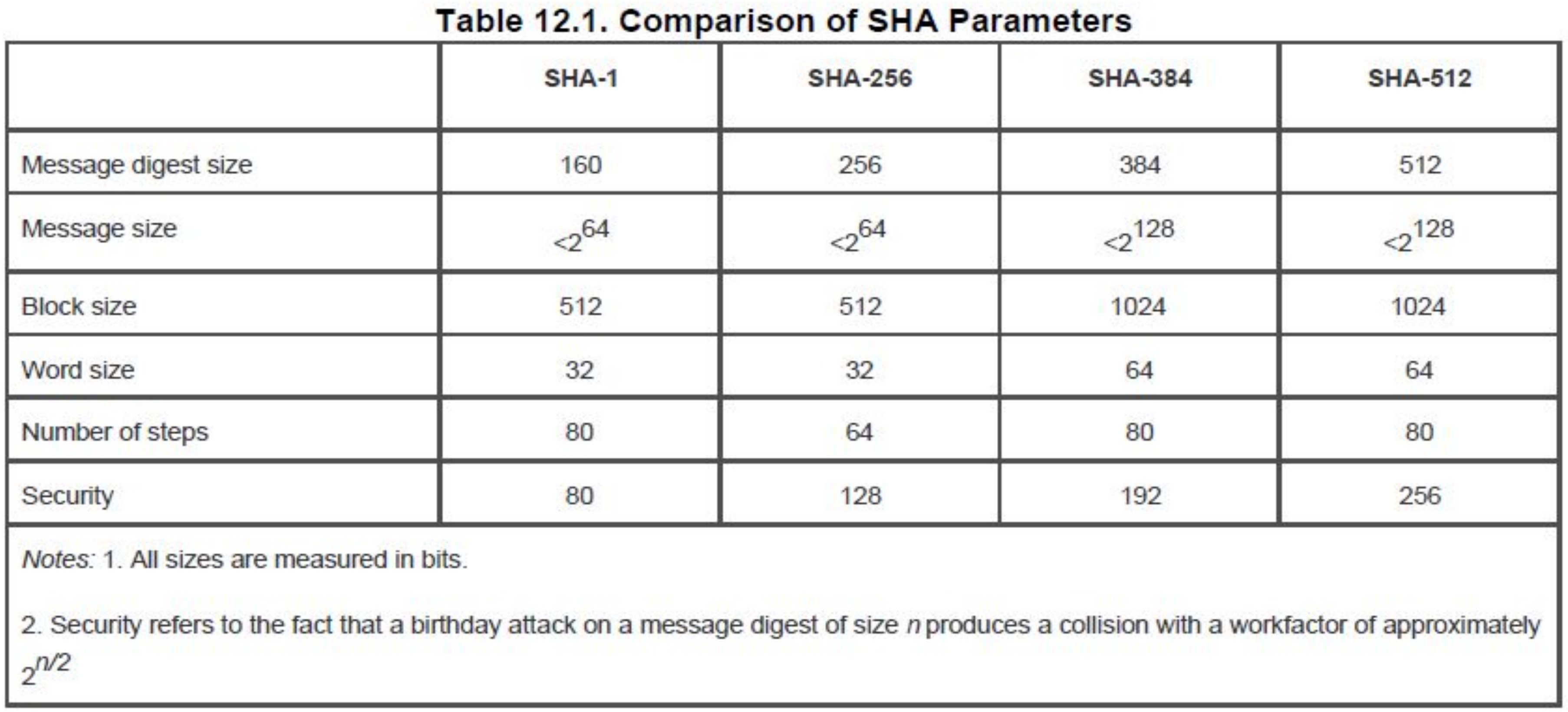


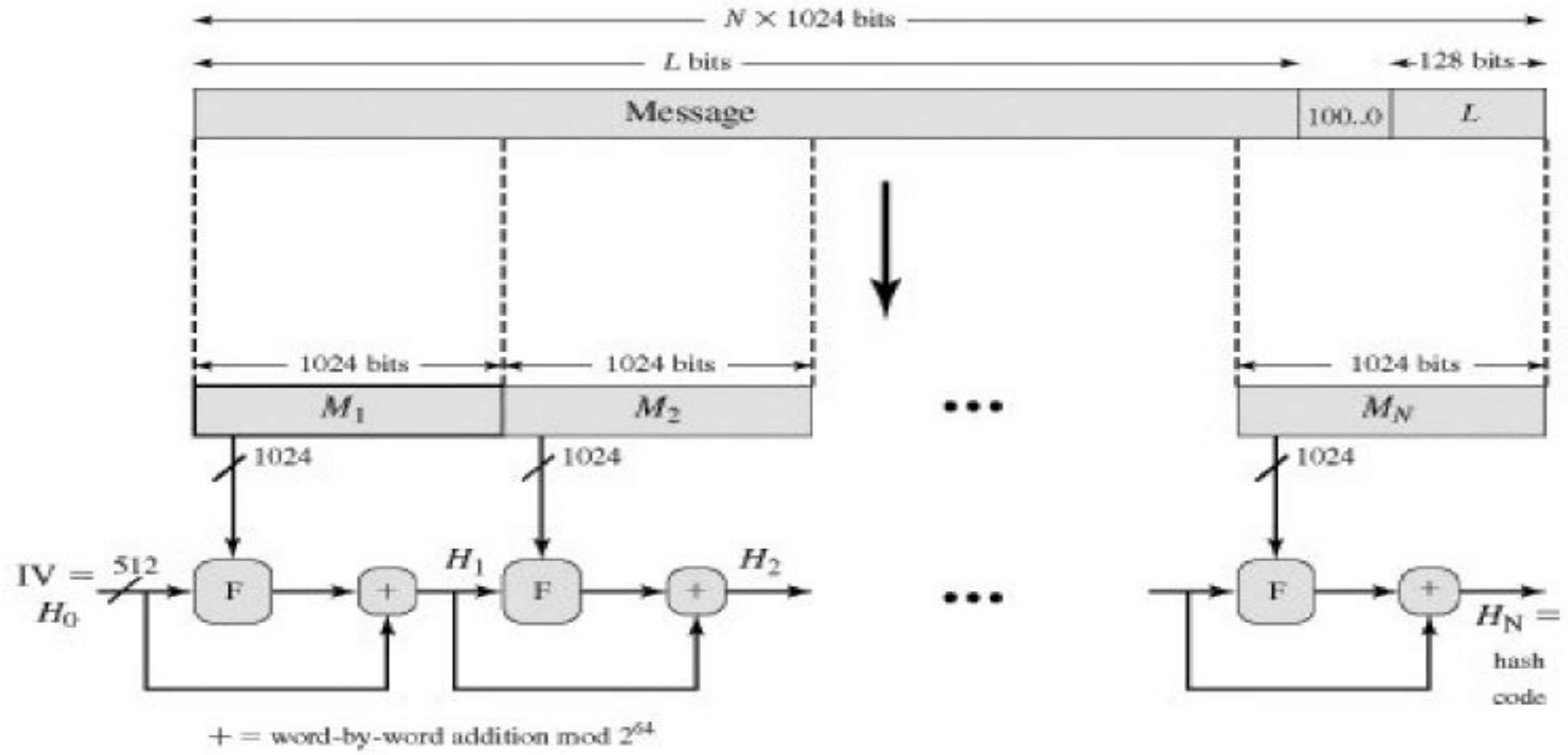
Table 12.1. Comparison of SHA Parameters

	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	80	64	80	80
Security	80	128	192	256

Notes: 1. All sizes are measured in bits.

2. Security refers to the fact that a birthday attack on a message digest of size *n* produces a collision with a workfactor of approximately $2^{n/2}$

Message Digest Generation using SHA-512



Message Digest Generation using SHA-512

- The algorithm takes as input a message with a maximum length of less than 2 ^{128} bits and produces as output a 512-bit message digest.

Message Digest Generation using SHA-512

- **Step 1. Append Padding bits:** Padding is always added even if the message is desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1-bit followed by the necessary number of 0-bits.
- **Step 2. Append length:** A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).
- **Step 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).

Message Digest Generation using SHA-512

Step 4: Process Message in 1024 bit (128-word) blocks: The heart of the algorithm is a module that consists of 80 rounds.

Step 5: After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512 bit message digest;

We can summarize the behavior of SHA-512 as follows:

H_0 = IV

H_i = SUM $_{64}$ (H_{i-1}, abcdefgh$_i$)

IV = Initial value,

MD=Message digest =

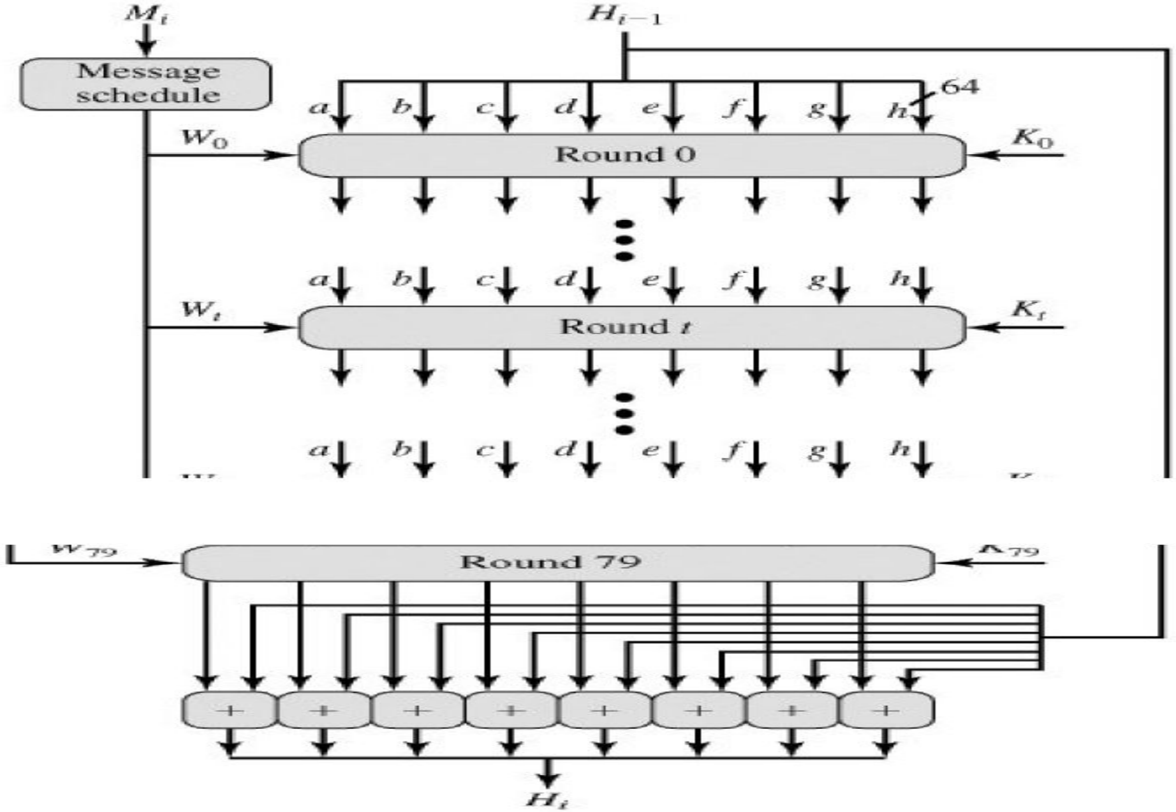
H_N , where N= no. of blocks in the messg.,

SUM $_{64}$ = Addition Module 2^{64} performed separately on each word of the pair of I/Ps

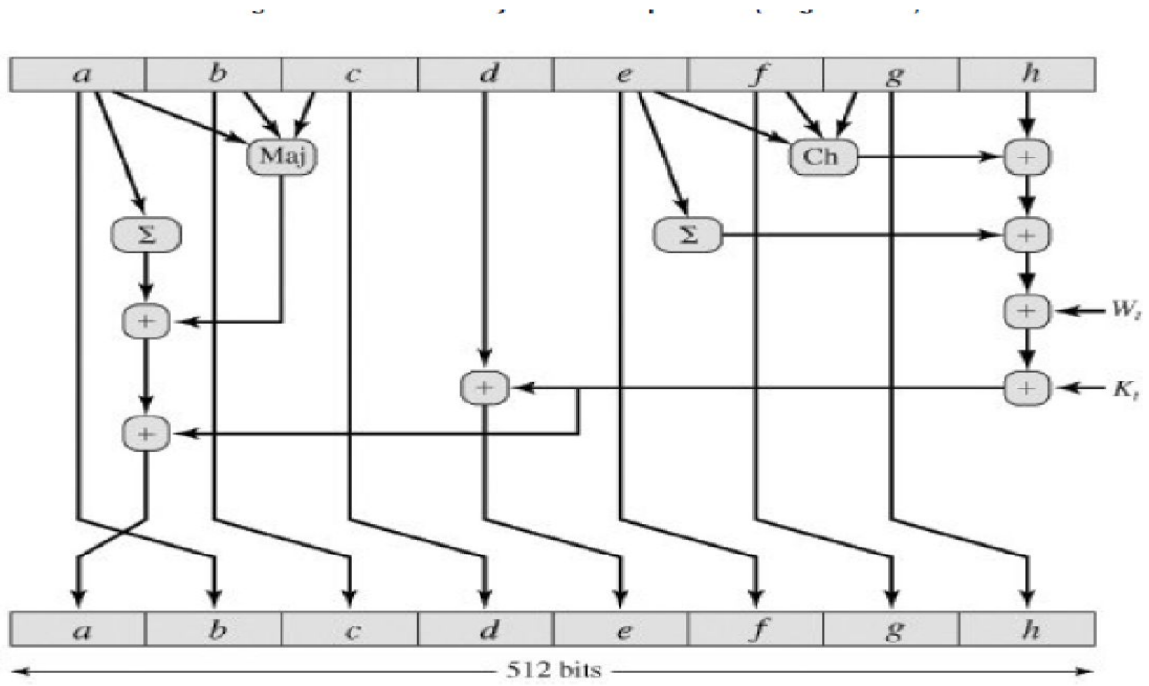
Message Digest Generation using SHA-512

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

SHA-512 Processing for a block



SHA processing for a round



SHA processing for a round

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

SHA processing for a round

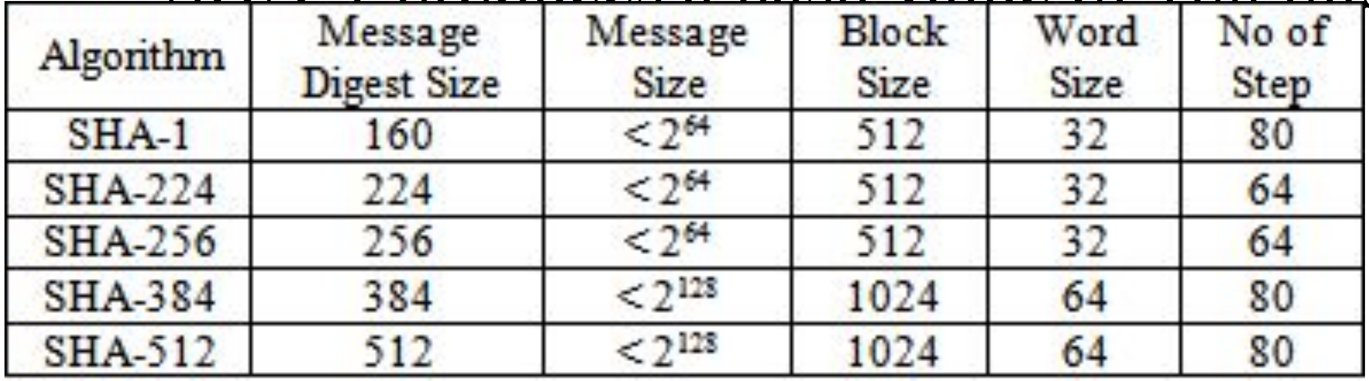
- where
- *T* = step number; 0 <*t* <79
- Ch(*e*, *f*, *g*)= (*e* AND *f*) XOR (NOT *e* AND *g*) *the conditional function*: *If e* then *f* else *g*
- Maj(*a*, *b*, *c*) = (*a* AND *b*) XOR(*a* AND *c*)XOR (*b* AND *c*) *the function is true only of the majority (two or three) of the arguments are true.*

$$\left(\sum_0^{512} a\right) = \mathrm{ROTR}^{28}(a) \oplus \mathrm{ROTR}^{34}(a) \oplus \mathrm{ROTR}^{39}(a)$$

- ROTR*n*(*x*) = circular right shift (rotation) of the 64-bit argument *x* by *n* bits
- *Wt* = a 64-bit word derived from the current 512-bit input block
- *Kt* = a 64-bit additive constant
- += addition modulo 2 64

Secure Hash algorithm(SHA)

- SHA is based on the hash function MD4, and its design closely models MD4.
- SHA-1 produces a hash value of 160 bits.



Algorithm	Message Digest Size	Message Size	Block Size	Word Size	No of Step
SHA-1	160	$<2^{64}$	512	32	80
SHA-224	224	$<2^{64}$	512	32	64
SHA-256	256	$<2^{64}$	512	32	64
SHA-384	384	$<2^{128}$	1024	64	80
SHA-512	512	$<2^{128}$	1024	64	80

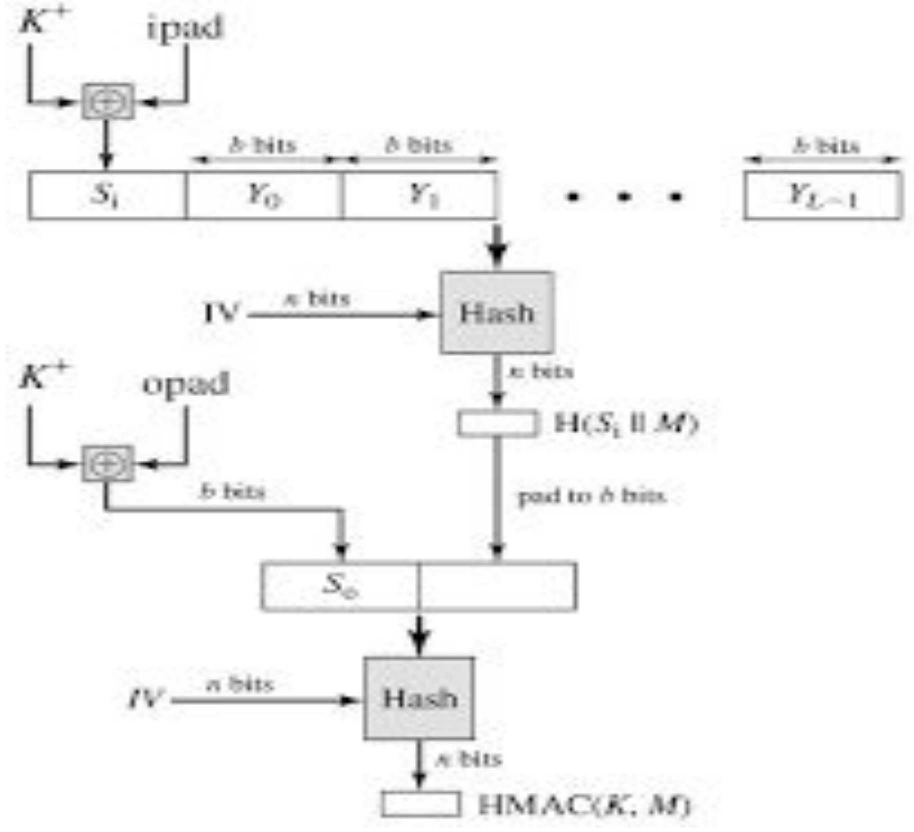
Message Digest (MD-5)

- Similar to SHA 1
- Faster than SHA 1 but SHA 1 Is more secure
- It Returns 128 bit message digest
- Takes an input of arbitrary length
- The input is processed in 512 bit blocks
- It is intended for digital signature applications
- The security is questionable now as with high speed processor the digest can be hacked by hackers

HMAC

- Hash-based message authentication code (HMAC) provides the server and the client each with a private key that is known only to that specific server and that specific client.
- The client creates a unique HMAC, or hash, per request to the server by hashing the request data with the private keys and sending it as part of a request.
- What makes HMAC more secure than Message Authentication Code is that the key and the message are hashed in separate steps.
- HMAC(key, msg) = H(mod1(key) || H(mod2(key) || msg))
- This ensures the process is not susceptible to extension attacks that add to the message and can cause elements of the key to be leaked as successive MACs are created.
- Once the server receives the request and regenerates its own unique HMAC, it compares the two HMACs. If they're equal, the client is trusted and the request is executed. This process is often called a secret handshake
- Ipad value is 00110110
- Opad value is 01011100

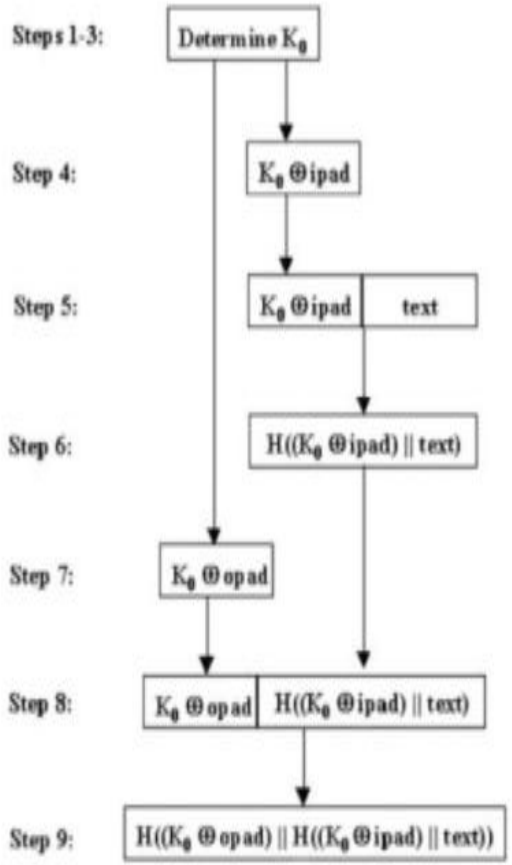
HMAC



K^+
ipad
b bits
S_i
Y_0
Y_1
Y_{L-1}
IV
n bits
Hash
$H(S_i \| M)$
K^+
opad
pad to b bits
S_o
HMAC(K, M)

Graphical representation of HMAC Algorithm

orithm



- Does Message encryption by itself provide a secure form of authentication. Justify
- Differentiate between encryption and hashing
-
-