Authentication Functions & Services

Security Requirements

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

③ Masquerade: Insertion of a message into the network from fraudulent source (a source that is intentionally false and meant to harm)

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 message between parties, including insertion, deletion and reordering

⑥ Timing modification: Delay or replay of message In a connection-oriented application, an entire session or sequence of a message could be replay of some previous valid session

(7) Source repudiation: Denial of transmission of message by source

(8) Destination repudiation: Denial of receipt of message by destination.

1 to 2 → encryption
2 to 6 → authentication
7 to 8 → digital signature

Message Authentication

Message authentication is concerned with
- protecting the integrity of message
- validating identity of originator
- non repudiation of origin

alternative functions used:
- message encryption
- message authentication code (MAC)
- hash function

MA functions

message encryption: Ciphertext of entire message serves as its authenticator

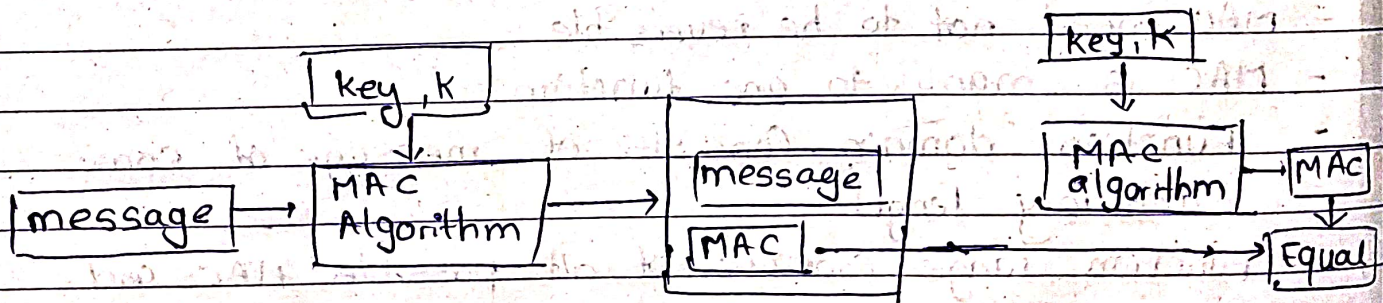
Message authentication code: a public function of the message and a secrete key that produces fixed length value serves as the authenticator

Hash Function: A public function that maps a message of any length into a fixed length hash value, which serves as the authenticator

Digital Signature: a public function that calculates a fixed length bit pattern or value of the message of any length involving keys, which serves as the authenticator

Message Authentication Code (MAC)

- MAC algorithm is a symmetric key cryptographic technique to provide message authentication
For establishing MAC process, the sender & receiver share a symmetric key



Sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value

Similar to hash, MAC function also Compresses an arbitrary long input into a fixed length output the major difference b/w hash & MAC is that MAC uses a secret key during compression

Sender forwards the message along with the MAC

on receipt of the message and the MAC, the receiver feeds the received message and the shared secret key k into the MAC algorithm and recomputes the MAC value.

receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match the it accepts.

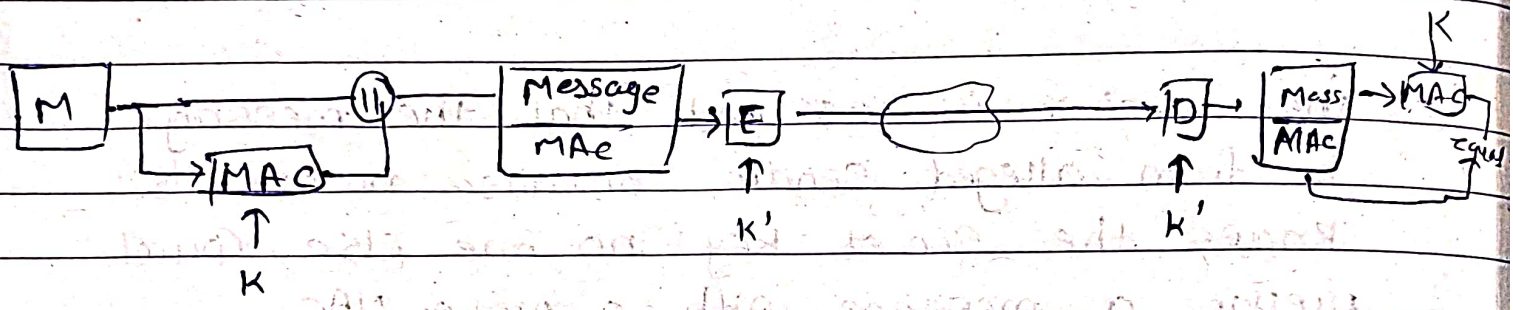
- MAC need not to be reversible
- MAC is many to one function
- Function domain consists of message of some arbitrary length
- Function range consists of all possible MAC's and all possible keys

n bit MAC: 2^n possible MAC's
k bit key: 2^k possible key

Internal & External Error Control (Code)

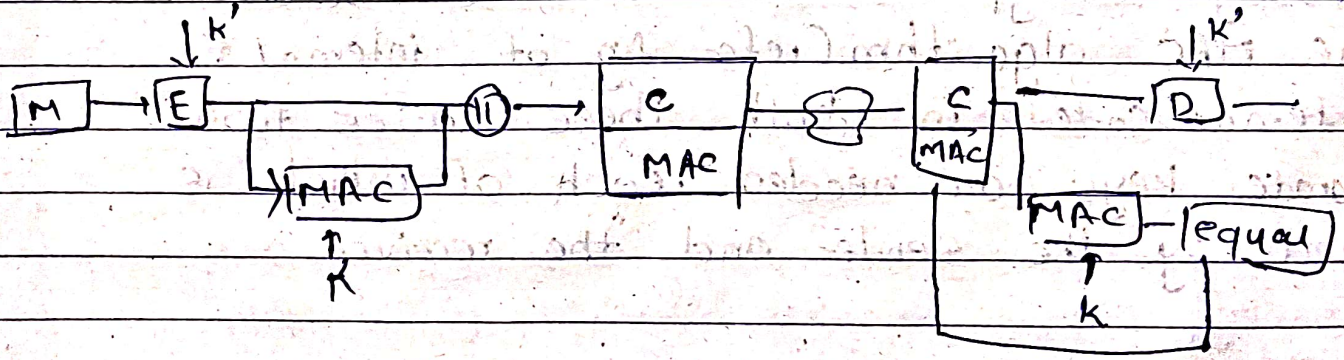
Internal error code

Sender encrypts the content before sending it through network for confidentiality. thus it provides confidentiality as well as authentication



External error code

When there is alteration in message, we decrypt it for waste to overcome that problem we opt for external error code : first apply MAC on encrypted message 'c' and compare it with received MAC value on the receiver's side and then decrypt 'c'



MAC

- 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.

- The receiver is assured that the message is from alleged sender. Because no one else knows the secret key. no one else could prepare a message with a proper MAC

- if the message includes a sequence no. then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence no.

Confidentiality can be provided by performing message encryption either after or before the MAC algorithm (refer fig of internal & external code) in both these cases two separate keys are needed, each of which is shared by the sender and the receiver.

Applications of MAC

(i) Same message is broadcast to a number of destinations. it is cheaper and more reliable to have only one destination responsible for monitoring authenticity.

(ii) when one side has a heavy load, cannot afford the time to decrypt all incoming messages. authentication is carried out on selective basis. messages being chosen at random for checking

(iii) authentication of computer program, always executed without wasting resource time in decryption

(iv) architectural flexibility due to separation of authentication (application layer) and confidentiality (transport layer)

(v) prolong period of ~~encryption~~ protection after decryption within home network.

#

- MAC provides confidentiality, it can also used for encryption secrecy.
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before.

- MAC is used because sometimes only authentication is needed

- Sometimes need authentication to persist longer than the encryption

MAC is not a digital signature

MAC is a cryptographic checksum

$MAC = C_K(M)$

- Condenses a variable-length message M
- using a secret key K
- to a fixed sized authenticator

is a many to one function
- potentially many messages have same MAC
- but finding these needs to be very difficult.

Requirements of MACs

① If an opponent observes M and $C_K(M)$, it should be computationally infeasible for the opponent to construct a message M' such that $C_K(M') = C_K(M)$ → even not knowing the key.

$C_K(M)$ → MAC generated by cryptographic function 'C' using key 'K' on message M

$C_K(M')$ → MAC generated by cryptographic function 'C' using key 'K' on message M'

if attacker doesnot know key than it is impossible to generate same message

(2) $C_K(M)$ should be uniformly distributed in the sense that for randomly choosen message, M and M' the probablity that $C_K(M) = C_K(M')$ is 2^{-n} where n is the no. of bits in the MAC → not knowing the key but having access to MAC.

the probablity that randomly chossen two messages $M \neq M'$ produces same MAC is very small even if attacker know MAC than also it is impossible to figure out message.

(3) Let M' be equal to some known transformation on M. That is $M' = f(M)$ eg f may involve inverting one or more specific bits in that case $pr[C_K(M) = C_K(M')] = 2^{-n}$ - known weak spots to match new with old MAC

even if attaker know funtion through which message is modified than also probablity of same message is very low i.e 2^{-n} and if it more than 2^{-n}.

Small change in message M creates a diffrent MAC completely. if it is producing same MAC then it is weak.

in encryption, the security mainly depends on the size of the key.

if the key is k bits, an attacker only need to try around 2^{k-1} attempts to break the encryption.

with MAC attacker would have access to both original message & its MAC, but if they don't have access to the key used to creat MAC.

if key size k is bigger than MAC size than it is possible to find key, attacker can try brute force search using multiple rounds.

How rounds works: (80 bit key & 32 bit MAC)

Round 1

attacker has the message M_1 and its MAC so they try all possible 2^k keys with M_1 to see which would produce the same key

out of 2^k keys only 2^{k-n} keys will match because they are fewer MAC values than keys.

$2^{80-32} = 2^{48}$ key matches

Round 2

attacker uses a second message M_2 and it's MAC to test only 2^{48} remaining keys

$2^{48-32} = 2^{16}$ match

Round 3

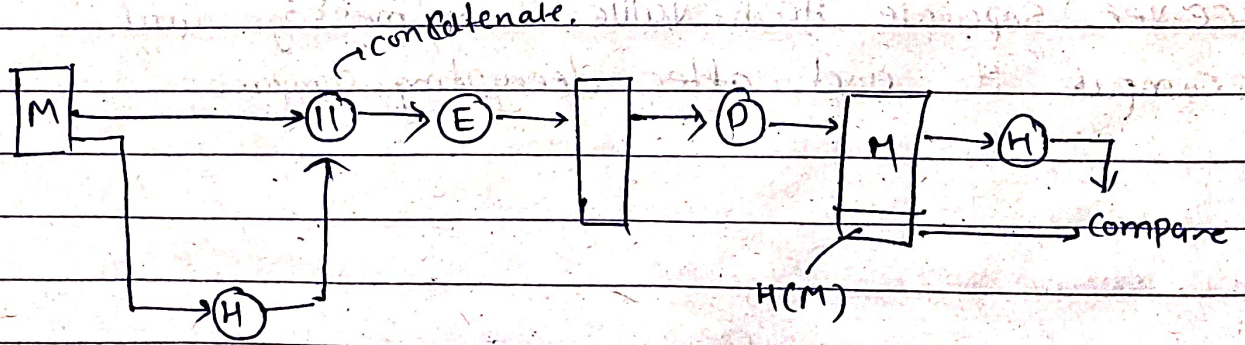
Finally, using a third message M_3 with its MAC, the attacker narrows down to about one key

$2^{16-32} = 1$

Hash Functions

- it Condenses arbitrary message to fixed size
- usually assume that the hash function is public and not keyed
- hash used to detect changes to message
- Can use in various ways with message
- most often to create a digital signature.

- a variation on the message authentication code is the one way hash function
- as with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed size output, referred to as a hash code H(M)
- unlike MAC, a hash code does not use a key but is a function only of the input message.

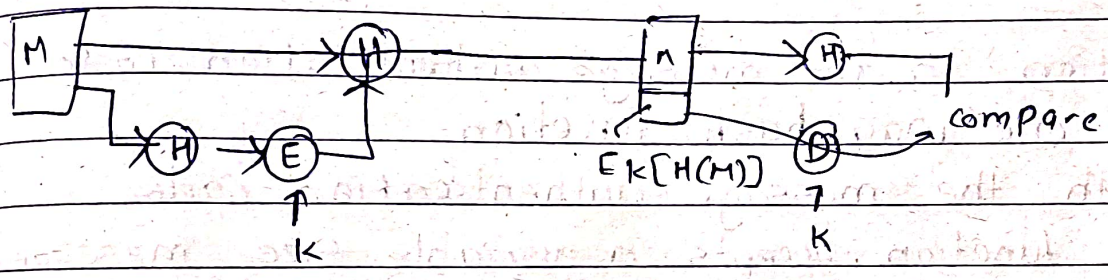


the message plus concatenated hash code is encrypted using symmetric encryption.

Secret key K is shared between A & B

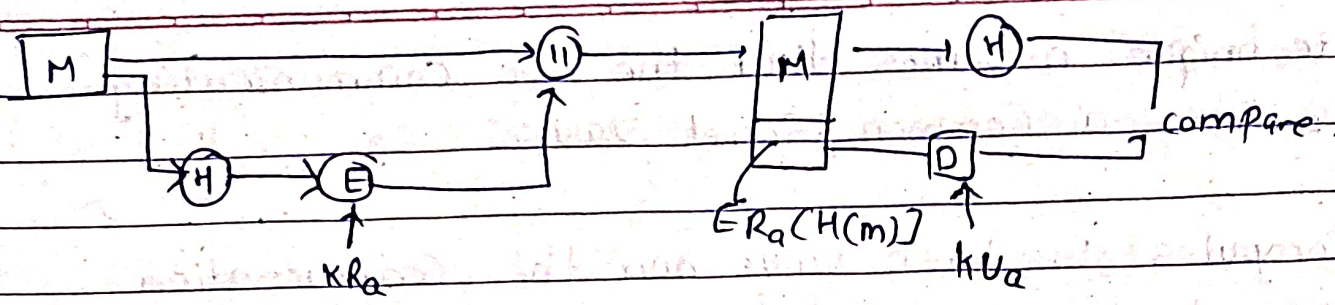
B decrypt the message separate concatenated hash & message recompute the hash value & compare.

because of encryption confidentiality is also provided.



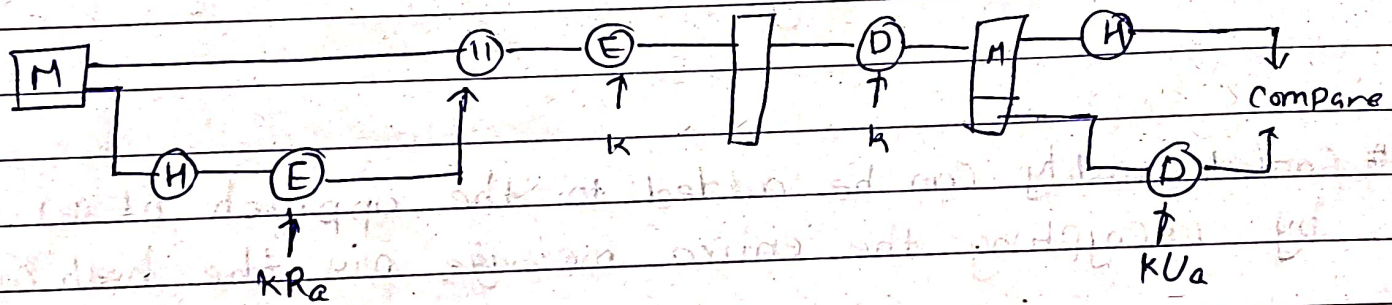
- only hash code is encrypted usmg symmetric encryption
- no confidentrality
- receiver separate Hash value from message and recompute it and after decrypting compare.

M — message
H → Hash function
E → Encryption

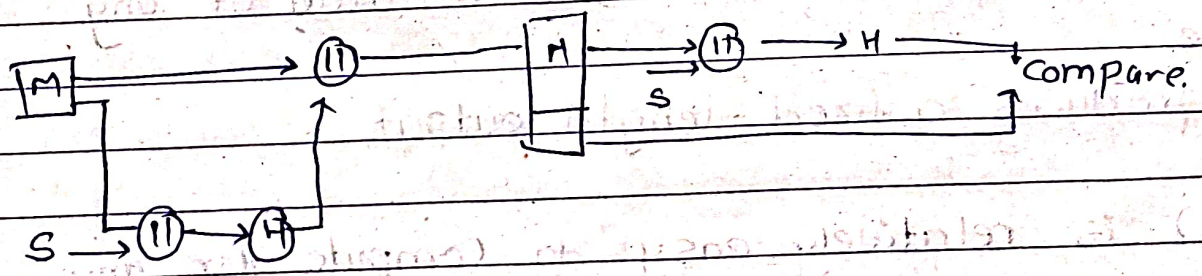


- only hash code is encrypted, using public key encryption and using the Sender's private key.

- provides authentication, it also provides a digital signature, because only the sender could have produce the encrypted hash codes



Confidentiality is added in previous function by encrypting whole message and concatenated encrypted hash.



this technique uses a hash function but no encryption for message authentication

the technique assumes that the two communicating parties share a common secret value S.

A computes the hash value over the concatenation of M and S and appends the resulting hash value to M.

Since B possesses S, it can recompute the hash value to verify.

Because the secret value itself is not sent, an opponent modify an intercepted message and cannot generate a false message.

Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

Requirements for Hash functions

(i) H can be applied to a block of data of any size

(ii) H produces a fixed-length output

(iii) H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical

(iv) for any given value h, it is computationally infeasible to find x such that H(x)=h, called as one way property

for a given o/p, h it is infeasible to find i/p x such that $H(x) = h$. (impossible to retrive i/p from o/p)

(v) For any given block x, it is Computationally infeasible to find $(y \neq x)$ with $H(y) = H(x)$
refered as Strong Collision resistance
for given i/p x it is impossible to find i/p y such that $H(y) = H(x)$

(vi) it is Computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
refered strong Colloision resistance

Simple Hash functions

$C_i = b_{i1} \oplus b_{i2} \oplus \dots b_{im}$

where
C_i = i^{th} bit of the hash code
m = number of n bit blocks in the input
b_{ij} = i^{th} bit in j^{th} block
$\oplus$ = XOR operation

- start with setting hash value to zero. it will be updateed as we process each block of data.

- data is divided into blocks of a fixed size each with n bits
- for each block
    a. rotate the Curcent hash value to the left by one bit
    b. XOR the block into the hash value

The rotation and XOR together help to randomize the hash value. rotation prevents patterns from emerging

Given a message consisting of a sequence of 64 bit blocks $X_1, X_2 \ldots X_N$ define the hash code C as the block-by-block XOR of all blocks and append the hash code as the final block

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

message is divided into blocks, where each block is 64 bits (e.g $X_1, X_2, \ldots X_N$)

to create a hash code C, take the XOR ($\oplus$) of each block

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

each bit in C is a combination of the corresponding bits in each block, making C a summary of the entire message

hash code is then appended to the end of the message as the last block X_{N+1}

last encrypt the entire message plus hash code using CBC mode to produce the encrypted message $Y_1, Y_2, \ldots Y_{N+1}$

Block Ciphers as hash functions

divide a message M into fixed size blocks $M_1, M_2 \ldots M_N$ and use a symmetric encryption system such as DES to compute the hash code G as follows

H_0 = initial value

$H_i = E_{M_i}[H_{i-1}]$

$G = H_N$

Similar to CBC but no secret key

Hash Functions & MAC Security