Chapter 11

Message integrity and Message Authentication

 # Message integrity :- In some cases we may not even need secrecy But we must need Integrity.

 # Document & fingerprint :- One way to preserve the integrity of a document is through the use of a fingerprint.

To ensure that the document has not been changed, Alice's fingerprint on the document can be compared to Alice's fingerprint on file. If they are not the same, the document not from Alice.

 # Document & fingerPrint :- One way to preserve the integrity of a document is through the use of a fingerprint.

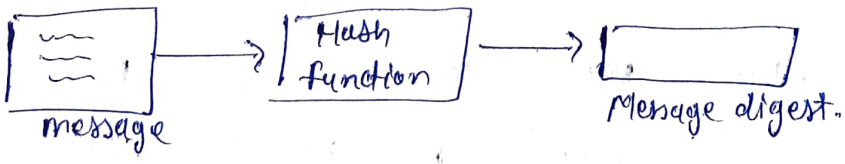
If Alice needs to be sure that the content of her document will not be changed, She can put her fingerprint at the bottom of the document. Eve cannot modify the contents of this document or create a false document because she cannot forge (copy) Alice's fingerprint.

To ~~ensure~~ ensure that the document has not been changed, Alice's fingerprint on the document can be compared to Alice's fingerprint on file. If they are not same, the document is not from Alice.

#Message and Message Digest :-

The ~~elect~~ document and fingerprint is a physical way of implementing intigrity. But for electronicaly or digitally implementation we have message and digest pair.

To preserve the integrity of a message, the message is passed through an algorithm called a cryptographic hash function. The function creates a compressed imege of the message that can be used like a fingerprint.



The message and message digest can be unlinked separatly. But it is not possible in case of document & fingerprint.

~~NOTE :- The mess~~

NOTE :- The message digest must be safe from change because of this we have to send a digest using a secure channel.

To check the integrity of message or document, we run the cryptographic hash function again and compare the new message digest with the previous one. If both are same, we are sure that the original message has not been changed.

Cryptographic Hash function Criteria :-

A Crytographic hash function must satisfy three criteria :-

1. Preimage resistance
2. Second Preimage resistance
3. Collision Resistance.

① **Preimage resistance :-** This ensures that given a hash value h, it should be computationally infeasible to find the original input 'x'.

$$h = f(x)$$

The goal is to prevent someone from reversing the hash back to the original input.

2. Second Preimage Resistance :-

Given the input x_1 it should be computationally infeasible to find a different input x_2 such that

$$H_1(x_1) = H_2(x_2) \quad \{H_1 \text{ or } H_2 \text{ may same or different}\}$$

It is like Locking a box - you can't find a different key that opens the same lock.

The goal is to insure integrity of data. If someone tries to replace original input with another input.

3. Collision Resistance :-

This ensures that it is computationally infeasible to find any two distinct input x_1 and x_2 such that

$$H(x_1) = H(x_2)$$

goal is to prevent the hash function from producing the same output from different inputs.

Message Authentication :- A message digest guarantees the integrity of message.. It guarantees that the message has not been changed.

But the message digest can-not authenticate the sender of the message.

The digest created by cryptographic hash function is normally called a "modification detection code" (MDC) The code can detect any modification in message.

for Message authentication (data origin authentication) is a message authentication code (MAC)

The MDC needs to be transferred through a safe channel. If we are sending MDC from Insecure channel then it can also be intercepted.

Message Authentication Code :-

To ensure the data origin authentication - that Alice is the originator of the message, not somebody else, we need to change MDC to MAC.

The difference is that the MAC include secret key between Alice and Bob, that can-not be possessed by Eve.

Alice uses a hash function that creates a MAC from the concatination of the key and message. $h(K|M)$.

She then sends a message to bob. over insecure channel.

Bob seperates the message from a MAC. & creates a new MAC & compare with the MAC which he received.

If the two MAC's are same then the message is authentic & Not modified.

Eve can see the message, but she can not copy a new message to replace it, because Eve can not posses the secret key b/w Alice & Bob. She is unable to create new MAC.

The MAC can be "prefix MAC" because the secret key is appended to the beginning of the message.

We can have a postfix MAC, in which the key is appended to the end of the message.

We can combine the prefix and postfix MAC, with the same key or two different key.

Security of MAC :-

Suppose Eve have intercepted message M & digest $h(K|M)$, then Eve can forge the message in following ways :-

(1) If the key size is small then she can use exhaustive search attack. She make all combination of key & corresponding hash & compare with the original one. If they matched with the original one then eve now know the key she can easily forge the message.

(2) If the key size is to big then she use preimage resistence. So that she can try to make the original message from hash. And Now she is able to found key. And able to forge message.

Nested MAC :- To improve the security of MAC "nested MACs" were designed. in which hashing is done in two steps:

In the first step key is concatenated with the message and hashed to create intermediate digest.

In the second step, the key is concatenated with the intermediate digest to create a final digest.

$h_1 \leftarrow H(K|M)$

$h_2 \leftarrow H(K|h_1)$

HMAC :- NIST has standared issue with nested MAC. So they reffered to HMAC.

HMAC is much more complex than the simplified nested MAC.

There is a additional feature such as a Padding.

The steps to create HMAC :-

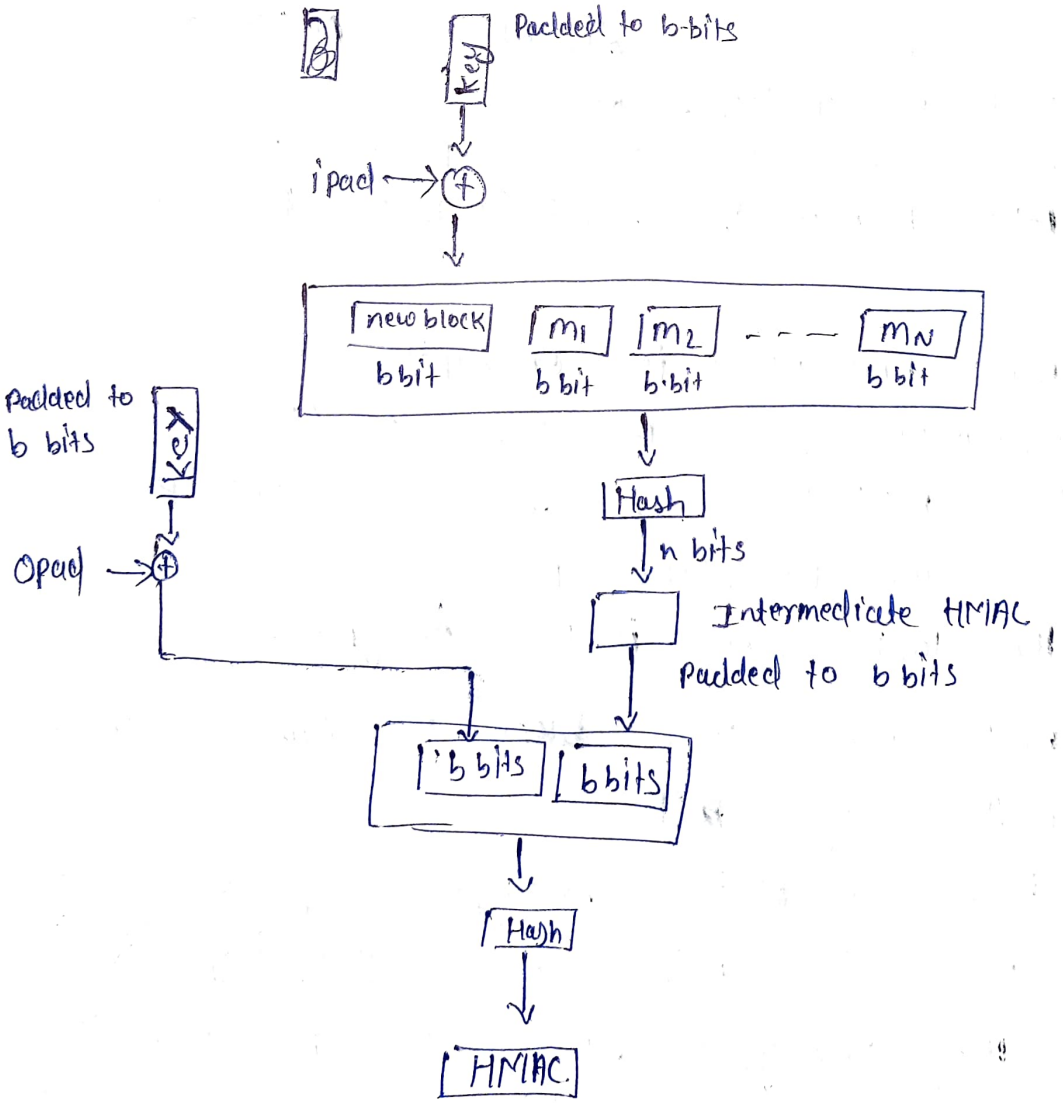
1. The message is divided into N blocks, each of b bits.
2. The secret key is left-padded with 0's to create b-bit key.
3. The resultant key is XORed with a constant called "ipad" (inpud pad) to create b-bit block. (new block)

   The value of ipad is the b/8 time repeatation of the sequences (00110110)(36).
4. The resultand block is added at front of N-block message to create, N+1 block.
5. Now the whole message (N+1 Blocks) are hashed to create n-bit intermediate digest.

(6) The ~~int~~ intermediate n-bit digest is padded with 0's to make b-bit block (left padded)

(7) The step 2 & 3 is repeated with different constant opad (output pad). The opad is b/8 time repeatation of 01011100 (5C)

(8) The result of step 7 is hashed with the same hashing algorithm to create final n-bit HMAC.



CMAC :- NIST defined new Algorithm for Data authentication. which is 'CMAC.

This Method is similar to the cipher block chaining (CBC) mode used for symmetric key enciphermenf.

But the goel is not encryption of Plaintext.

The Idea is to create one block of MAC from N blocks of plaintext using Symmetric key cipher N times

(1) The Message is divided into N blocks, each m bit long. If the last block is Not m-bits then it is padded with a 1-bit followed by enough 0-bits to make it m-bits.

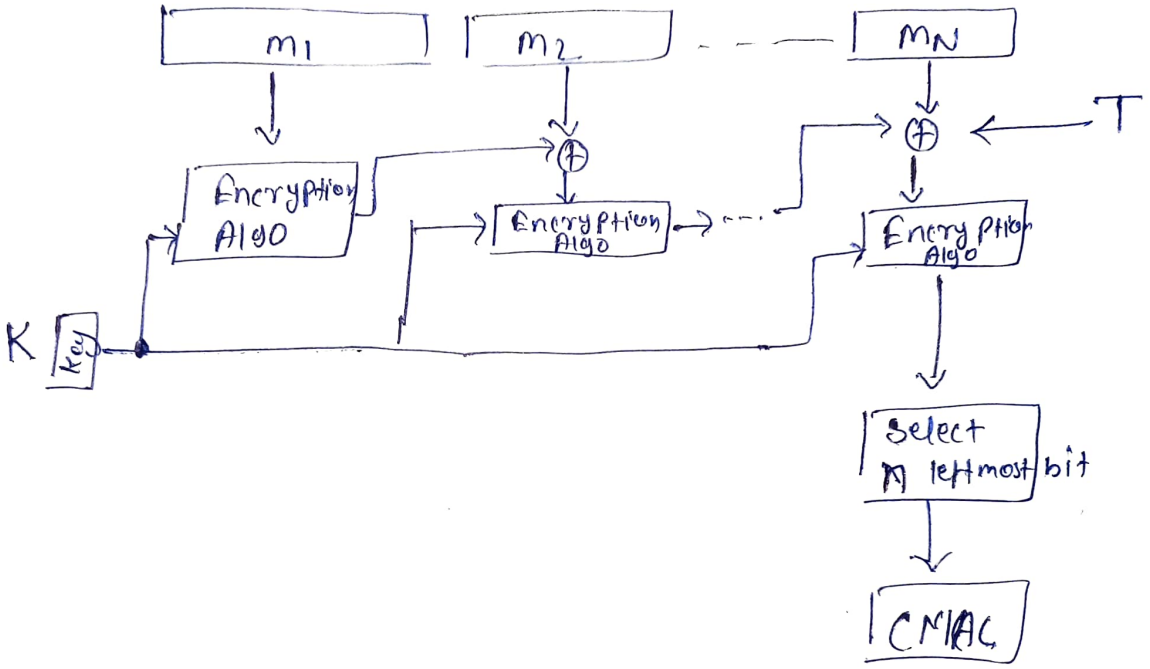
(2) The first block is encrypted with a Symmetric key. This block is XORed with the next block & result is encrypted with a same key. The proces continue till the last block is Not encrypted.

(3) But the last block is encypted with another key K.T

This key KT is encryption of ~~m-bit 0's~~ m, 0-bits using the original key K

~~the~~ encryption of 0-bit is multiplied with 'n' if padding at last block is applied else multiplied with 'n²'

④ After the ~~multiplication~~ XoRed of key T at last block the n leftmost bit is selected this leftmost bits are CMAC.



T.

How T is created :-