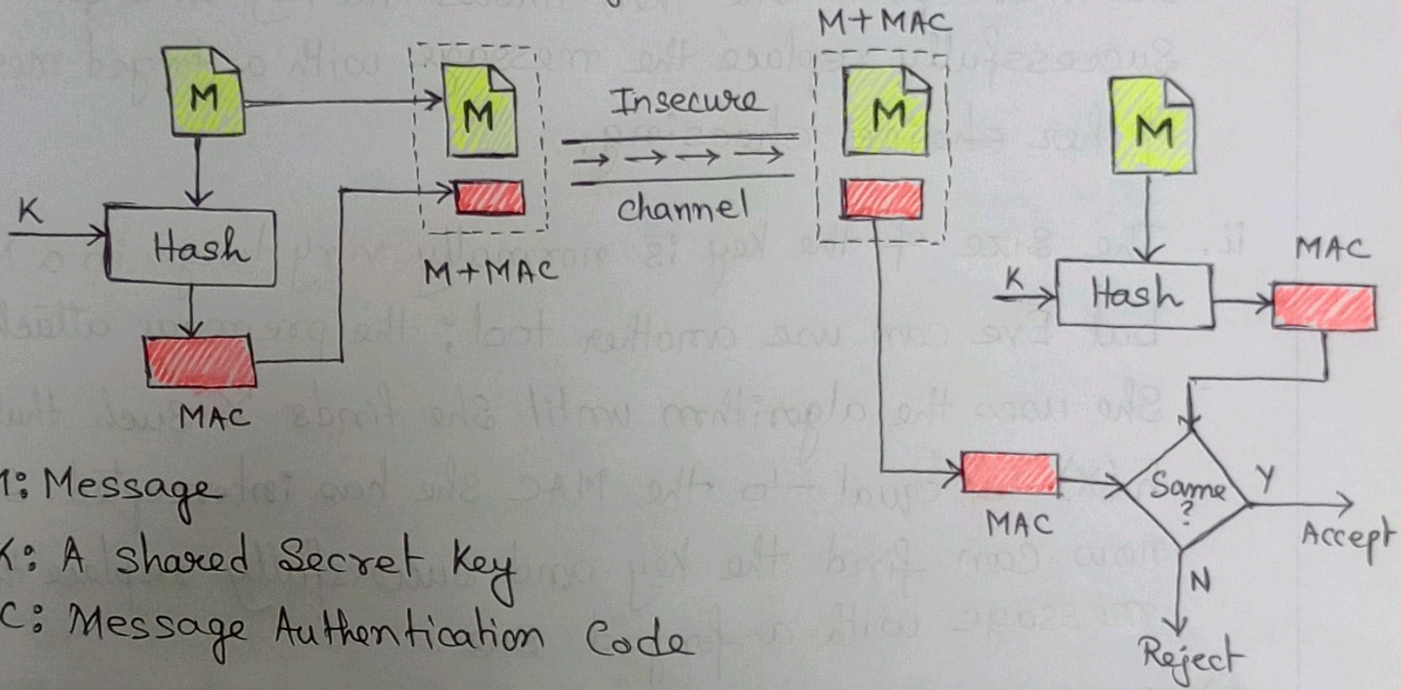
Message Authentication Code (MAC): To ensure the intigrity of the message and the data, that Alice is the originator of the message, not somebody else we need to use message authentication code (MAC). A MAC also known as a cryptographic checksum, is generated by the function c of the form, $T = MAC(K, M)$

where M is a variable length message, K is a secret key and MAC(K,M) is the fixed length authenticator



M: Message
K: A shared Secret Key
MAC: Message Authentication Code

Alice uses a hash function to create MAC with the key and the message. She sends the message and the MAC to Bob. Then Bob seperates the message from the MAC. Bob then makes a new MAC from the shared secret key and message. Bob now compares the newly created MAC with the one received. If the two MACs matches, the message is authentic and has not been modified by an adversary.

Security of MAC:

Suppose Eve (the adversary) has intercepted the message M and MAC. How can Eve forge a new message without knowing the secret key?

There are three possiable cases,

i. If the size of the key allows exhaustive search, Eve may prepare all possible keys at the begining of the message and make a digest, to find the digest equal to one intercepted. She then knows the key and can successfully replace the message with a forged message of her ~~choice~~ choosing.

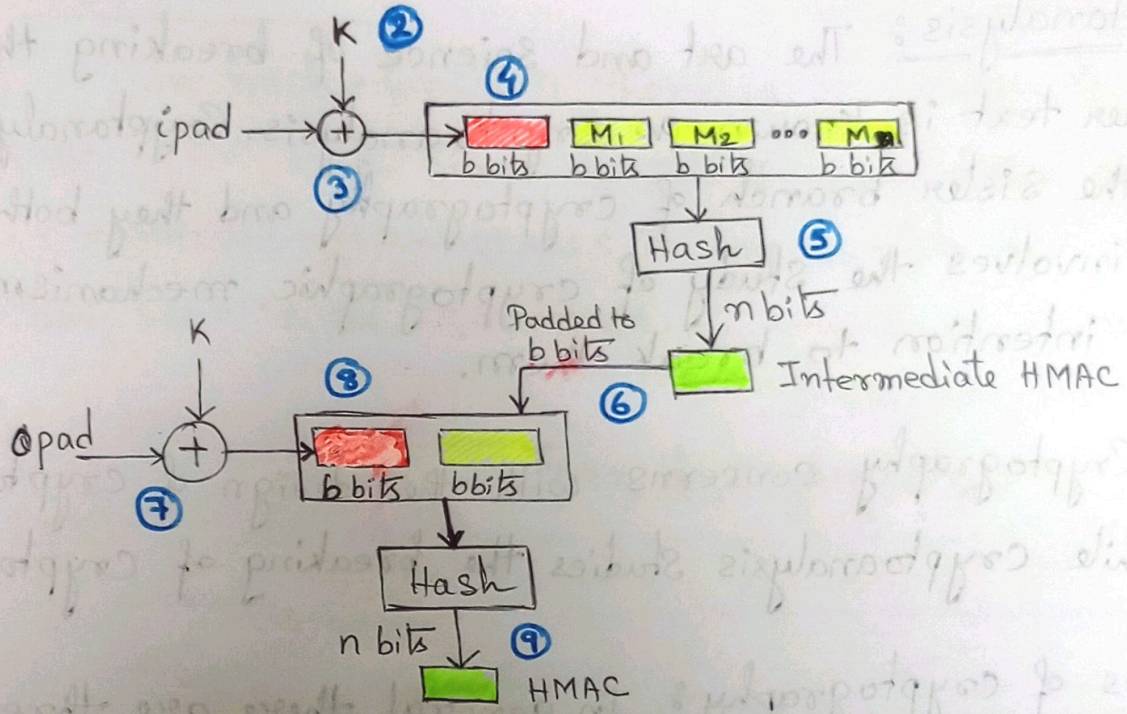
ii. The size of the key is normally very large in a MAC, but Eve can use another tool; the preimage attack. She uses the algorithm until she finds X such that $h(X)$ is equal to the MAC she has intercepted. She now can find the key and successfully replace the message with a forged one.

iii) Given some pairs of message and their MAC's, Eve can manipulate them to come up with a new message and its MAC

Note: The security of MAC depends on the security of the underlying hash algorithm.

Hash-based message authentication code (HMAC):

HMAC is a type of message authentication code (MAC) that is acquired by executing an cryptographic hash function on the data that is to be authenticated and a secret key. HMAC is used for both data intigrity and authentication.



1. The message is divided into N-blocks, each of b bits.
2. The Secret key is left padded with zeros to create a b-bit key
3. The result of step 2 is XORed with a constant called ipad (input pad) to create a b bits block.
4. The resultant block is prepended to the N-block message.
5. The result of step 4 is hashed to create [redacted] an n-bit digest.
6. The intermediate n-bit HMAC is left padded with 0 to make b bits block.
7. Step 2 and step 3 [redacted] repeated by a different constant opad
8. The result of step 7 is prepended to the block of step 6
9. The result of step 8 is hashed with the same hashing algorithm to create the final n-bit HMAC.