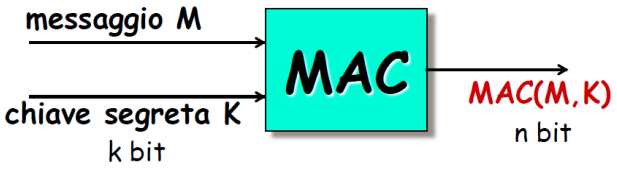
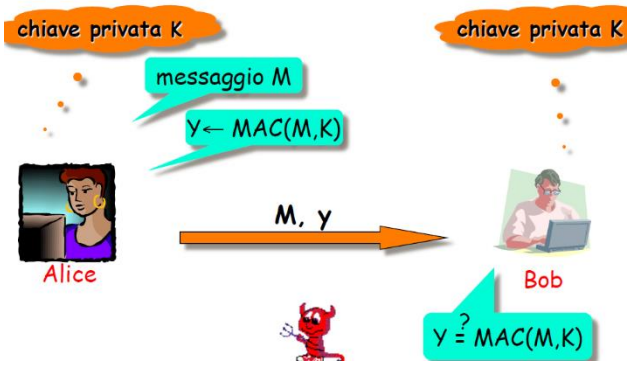


Il MAC è una primitiva totalmente deterministica che prende in input un messaggio M ed una chiave segreta K e dà in output un codice di autenticazione di quel messaggio.

Viene usato in applicazioni in cui si vuole essere sicuri che il messaggio è autentico (cioè che lo riesco ad autenticare con la chiave segreta) e che sia integro. Dal punto di vista concreto:

Se Alice deve inviare un messaggio a Bob, entrambi condividono una chiave privata K e Alice manda la coppia messaggio **M** e **y**= MAC(M,K). Quando Bob riceve queste 2 informazioni ricalcola il MAC sul messaggio M utilizzando la chiave K condivisa. Se il MAC che ha generato è lo stesso di quello che gli è stato mandato allora il messaggio è autentico, cioè viene certificato che il messaggio è stato mandato da Alice ed è integro (nessuna manipolazione del contenuto).



Questo concetto è molto più debole rispetto ad una firma digitale perché la chiave privata la posseggono sia Alice che Bob e nessun'altro possiede questa chiave privata, quindi non è opponibile rispetto ad altre persone il fatto che sia stato calcolato il codice di autenticazione **y** . In genere il MAC è molto veloce da realizzare ed è molto più debole della firma digitale.

Il MAC, come descritto in precedenza, serve solo per autenticare qualcosa, ma non a cifrare. Infatti nell'esempio precedente Alice manda a Bob il messaggio in chiaro (non è cifrato). Se un attaccante osserva la transizione che avviene tra Alice e Bob ha la possibilità di leggere il testo in chiaro.

Se si volesse avere anche confidenzialità, il messaggio che viene mandato deve essere modificato in modo opportuno e ciò può essere fatto in due modi:

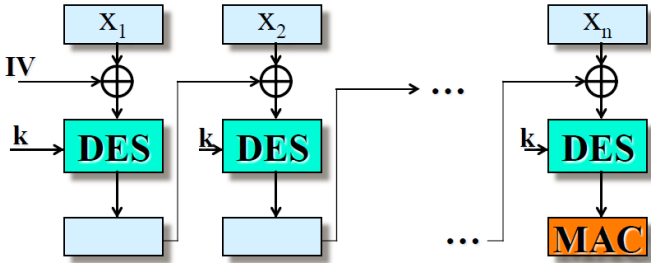
- Cifro il messaggio con una nuova chiave K' condivisa da Alice e Bob, in modo tale che il MAC del messaggio viene ottenuto con il testo cifrato (e quindi no quello in chiaro come visto precedentemente) e la chiave K → $E_{K'}(M)$, $MAC(E_{K'}(M),K)$.
- Utilizzo una nuova chiave K' per cifrare tutto insieme, messaggio e MAC, in modo tale che se un attaccante origlia il passaggio di informazioni, vedrà un solo testo cifrato senza poter leggere alcuna informazione in chiaro. → $E_{K'}(M,MAC(M,K))$.

Entrambe le opzioni sono valide per ottenere confidenzialità. (Si ricordi che la confidenzialità non è l'obiettivo del MAC, in quanto i suoi obiettivi sono autenticità ed integrità).

Ci sono varie tecniche per ottenere il MAC basate su 2 grosse famiglie: cifrari a blocchi (Data Authentication Algorithm: CBC-MAC) e funzioni Hash (HMAC).

CBC-MAC:

La modalità di cifratura (**C**ipher **B**lock **C**haining) è la seguente: il messaggio viene diviso in blocchi di 64 bit e viene cifrato di volta in volta utilizzando sempre la stessa chiave. Una volta eseguita questa procedura, il MAC risultante è l'ultimo blocco della cifratura, oppure un valore troncato: da 16 a 64 bit più a sinistra. Questa modalità è poco sicura ed è stata illustrata per fornire una semplice idea generale.



Più interessante e più utilizzato e il MAC basato su funzione Hash che vengono usate come black-box:

HMAC

Si può utilizzare qualsiasi funzione Hash che viene iterata con inizialization vector IV (MD5, SHA-1, RIPEMD-160…) e utilizza due costanti specifiche:

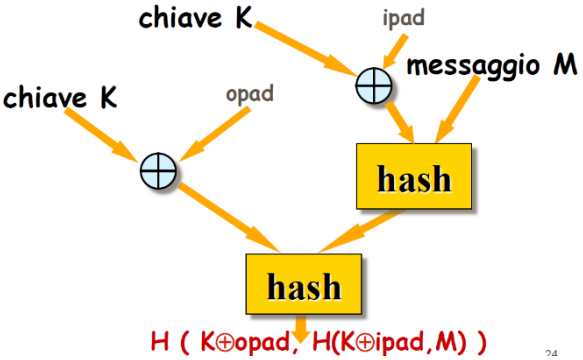
- **ipad**= byte 00110110 (36 in esadecimale) ripetuto b/8 volte.
- **opad**= byte 01011100 (5C in esadecimale) ripetuto b/8 volte.

Queste costanti vengono usate per avere lunghezze di blocchi precise in base alla funzione Hash utilizzata.

La struttura dell'HMAC è la seguente:

Partendo da sinistra, ho la chiave K e ne devo fare lo XOR bit a bit con la costante **opad** e il risultato è il primo input dell'Hash posto in basso. La stessa chiave K entra in input ad uno XOR con l'**ipad** e il risultato viene concatenato al messaggio M e di tutto ciò (XOR(K,ipad),M) ne calcolo l'Hash, il cui risultato andrà in input all'Hash posto in basso, in particolare entra come secondo input. Dunque l'ultimo Hash è il risultato di:

H(K⊕opad, H(K⊕ipad, M)).



La chiave K può generare 2 problemi: poiché il valore di **opad** è lungo 512 o 1024 bit a seconda della funzione Hash che è stata scelta, devo garantire che la chiave K abbia la stessa lunghezza della costanti (poiché si effettua uno XOR bit a bit). Se ad esempio le lunghezze di **opad** e **ipad** sono di 512 bit e la chiave non ha lunghezza 512 bit, devo in qualche modo eguagliarli:

- Se la lunghezza della chiave è più piccola, faccio un padding (cioè metto una serie di 0), in modo tale che le lunghezza coincidano.
- Se la lunghezza della chiave è più grande, viene effettuato un Hash della chiave K (si ricordi che l'output dell'Hash è un numero fissato di bit) e, se richiesto, fare un padding.

Diverse volte si usano solo i primi t bit dell'Hash (Output troncato). Ad esempio:

- HMAC-SHA1-**80** (solo i primi 80 dei 160 bit).
- HMAC-MD5 (tutti i 128 bit).
- HMAC-SHA256 (tutti i 256 bit).

È raccomandato che comunque questo t troncato debba essere $\geq n/2$ per una funzione Hash di n bit.

Se si vuole avere confidenzialità con queste idee spiegate ci sono varie modalità utilizzate a seconda di vari casi che si basano 2 chiavi: k_1 e k_2. Tuttavia non illustreremo queste idee, basta solo sapere che si mischia cifratura e MAC in tantissimi modi.

Il NIST ha approvato diversi modi per combinare MAC e confidenzialità, tra cui CCM e GCM. Anche in questo caso viene solo illustrato senza fornire ulteriori dettagli pratici.

Un esempio pratico del MAC si ha quando un'emittente televisiva trasmette nelle case degli spettatori partite di calcio. Tutte le informazioni che l'emittente invia sulle schede degli utenti, devono essere autenticate. La scheda quando riceve un comando, questi comandi sono legati a codici MAC.