# Hash Crittografici

# Outline

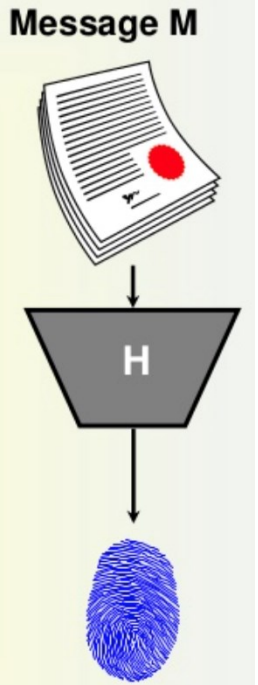
- Funzioni Hash
- Funzioni Hash Crittografiche
- Autenticazione dei messaggi
- Firma digitale e altre applicazioni

# Outline

- Funzioni Hash
- Funzioni Hash Crittografiche
- Autenticazione dei messaggi
- Firma digitale e altre applicazioni

# Funzioni Hash

- La funzione hash:
  - prende in input un blocco M di lunghezza variabile
  - genera un messaggio di impronta digitale a lunghezza fissa h = H(M).
- Non viene utilizzata alcuna chiave



# Outline

- Funzioni Hash
- Funzioni Hash Crittografiche
- Autenticazione dei messaggi
- Firma digitale e altre applicazioni

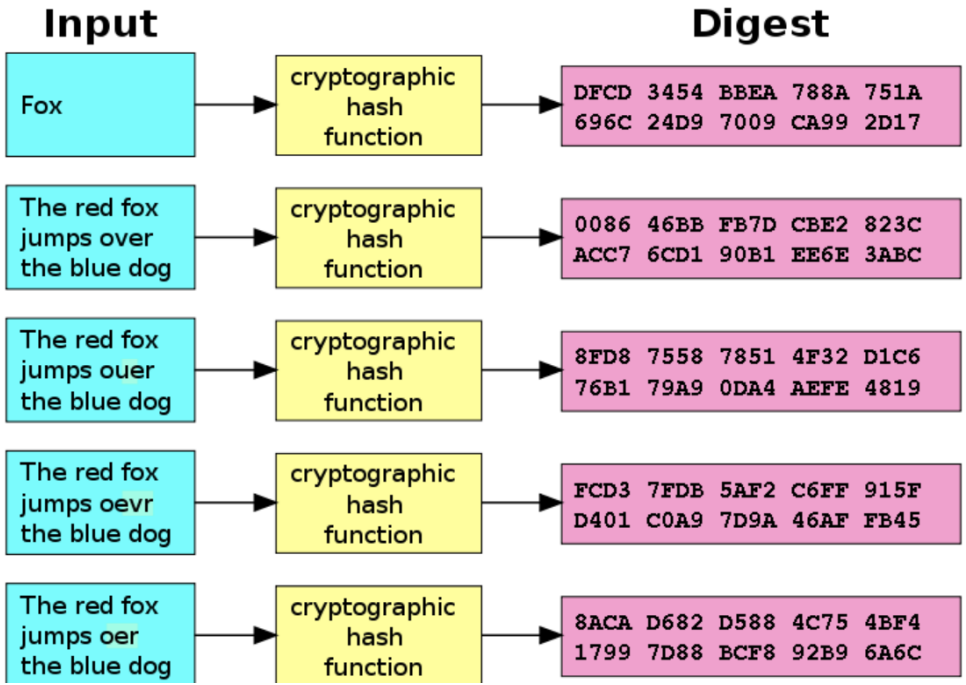
# Funzioni hash crittografiche

- Le funzioni hash di interesse per la sicurezza informatica sono le Funzioni hash crittografiche, cioè quei membri di una famiglia di funzioni che sono :
  - One-way (non-invertibili)
  - Collision resistant
- Per essere efficaci le funzioni di hash crittografico devono evitare che due messaggi siano associati allo stesso hash e garantire la resistenza alla falsificazione del messaggio.
- N.B.: Non necessariamente si usa la crittografia...

# Resistenza di Hashing crittografici

- Preimage resistance (One-way function): Dato h = H(m) , non è possibile derivare m conoscendo solo h (trovare x tale che H(x) = m)
- 2nd preimage resistance (Weak collision resistance): Dato h = H(m1) e m1, non è computazionalmente possibile trovare m2 ≠ m1 in modo tale che H(m1) = H(m2).
- Strong collision resistance: deve essere computazionalmente impraticabile trovare una coppia m1 e m2 tale che H(m1) = H(m2).
- Weak collision resistance è legata a uno specifico input, mente strong collision resistance si applica per due input arbitrari.

# Hashing crittografico: un esempio




- Una funzione di hash crittografico all'opera: Un piccolo cambiamento nell'input ("over") cambia molto l'output.
- Effetto valanga se un input viene modificato leggermente (anche di un solo bit), l'output cambia in modo significativo (metà dei bit di output).

# Secure Hash Algorithm - SHA

- Ci sono "standard" per le funzioni di hash crittografico, come lo era il DES, e ora AES, per la crittografia simmetrica.
- Lo standard per le funzioni crittografiche è chiamato Secure Hash Algorithm (SHA), una suite di funzioni crittografiche sviluppata dall'NSA dal 1993.
- Due famiglie principali:
  - SHA-1: restituisce un Message Digest (MD) di 160 bit
  - SHA-2: consiste di sei funzioni hash che si differenziano per la lunghezza del digest che è 224, 256, 384 or 512 bit: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256.

# SHA-1 uso

SHA-1 è stato ampiamente utilizzato in :

- Molti protocolli come TLS, SSL, PGP, e SSH.
- Alcuni servizi di versioning come Git e Mercurial, che permettono alle persone di modificare gli stessi file su un server remoto e sincronizzare le modifiche evitando conflitti e trasferimenti di file non necessari.

# Punti deboli di SHA-1

- SHA-1 restituisce un Message Digest (MD) di 160 bit, tipicamente reso come numero esadecimale, lungo 40 cifre. È stato progettato dalla United States National Security Agency.
- SHA-1 risale al 1995 e dal 2005 è noto che esso sia vulnerabile.
- A partire dal 2010 molte organizzazioni ne hanno raccomandato la sostituzione.
- Tutti i principali browser web hanno cessato di accettare i certificati SSL basati su SHA-1 nel 2017.
- A partire dal 2020, si raccomanda di utilizzare sempre le varianti SHA-x.

# SHA-2

- La famiglia SHA-2 è composta da sei funzioni hash con digest (valori di hash) che sono 224, 256, 384 o 512 bit :
  - SHA-256 e SHA-512 utilizzano *shifting* e costanti additive diverse, ma le loro strutture sono praticamente identiche, differendo solo per il numero di *round*.
  - SHA-224 e SHA-384 sono versioni troncate di SHA-256 e SHA-512, calcolate con diversi valori iniziali.
  - SHA-512/224 e SHA-512/256 sono anche versioni troncate di SHA-512, ma i valori iniziali sono generati in modo diverso.
- SHA-2 è stata definita dalla NIST statunitense.
- Gli algoritmi sono brevettati, ma gli Stati Uniti hanno rilasciato il brevetto con licenza royalty-free.

# Outline

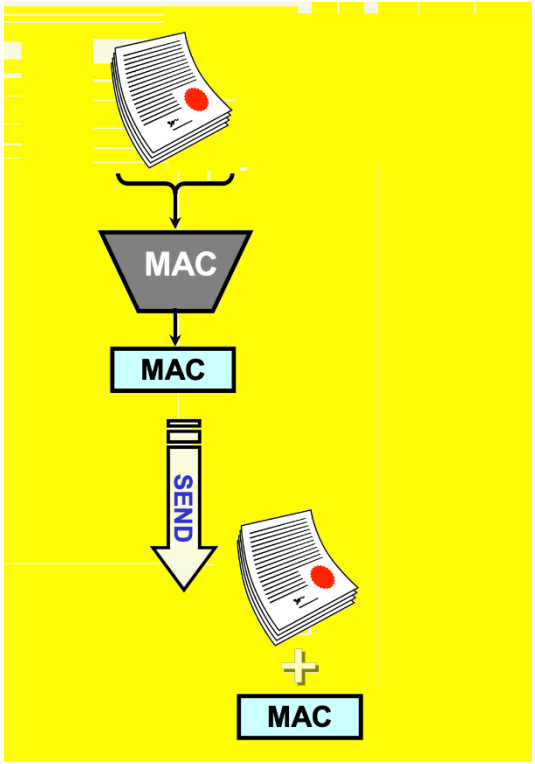
- Funzioni Hash
- Funzioni Hash Crittografiche
- Autenticazione dei messaggi
- Firma digitale e altre applicazioni

# Funzioni Hash in uso

- L'utilizzo principale delle funzioni hash è quello di garantire l'integrità di un messaggio M.
  - Il messaggio M viene inviato insieme al risultato della funzione hash applicata a esso.
  - Nel caso in cui M venga cambiato nel percorso per diventare M', il ricevente, calcolando la funzione hash su M', otterrà, con una probabilità molto alta, un valore diverso da quello inviato insieme a M.

# Autenticazione dei messaggi

- Il cosiddetto software *Message Authentication Codes* (MAC) è utilizzato per produrre un digest dei messaggi (Message Digest MD - MAC) da un messaggio M.
- Una volta che un MAC ha prodotto un MD, la coppia (M, MD) viene inviata per un successivo controllo.



# Autenticazione dei messaggi

- Il mittente A calcola l'hash del suo messaggio, lo allega al messaggio stesso e lo invia a B.
- Il destinatario B, al ricevimento, separa il messaggio dall'hash e ricalcola l'hash.
- A questo punto, il confronto tra l'hash calcolato e l'hash ricevuto permette al destinatario di verificare se il messaggio è intatto (cioè l'hash corrisponde).
- Se gli hash non coincidono, il messaggio può provenire da un'altra fonte o può essere stato modificato durante il viaggio.

# Autenticazione cifrata dei messaggi

- Viene utilizzata una chiave segreta condivisa che potrebbe essere generata tramite Diffie-Hellman.
- Le funzioni di hash crittografico garantiscono l'integrità.
- Se il MAC è cifrato, l'autenticità è garantita.
- Se il messagio (M) è cifrato, la confidenzialità è garantita.
- Ovviamente, sia M che MD possono essere criptati per garantire autenticità e confidenzialità.

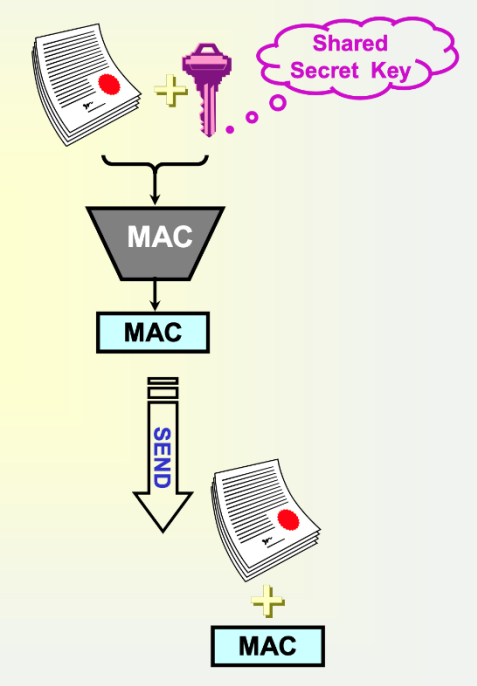
# Autenticazione cifrata dei messaggi

- Se A e B condividono una chiave segreta k, per un cifrario simmetrico, allora A può cifrare M o MD o entrambi e inviarli a B:
  - Se B riesce a decifrare M, allora può considerare il messaggio come autentico (può supporre che provenga effettivamente da A, in quanto A è l'unico che potrebbe aver criptato M con la chiave condivisa);
  - Se B calcola l'hash di M con la stessa funzione utilizzata da A e il risultato è uguale all'MD ricevuto, allora B può assumere l'integrità di M.
  - Se M non è confidenziale, per risparmiare le risorse di calcolo necessarie per cifrare un intero messaggio (che può essere di lunghezza arbitraria), A può cifrare solo MD.

# Funzioni di autenticazione

- Per costruire una buona funzione MAC, è opportuno usare le funzioni di hash crittografiche:
  - Le funzioni Hash come SHA sono computazionalmente più veloci della crittografia simmetrica e sono sufficienti per verificare l'integrità di un messaggio.
  - Algoritmi di crittografia simmetrici genererebbero dei MAC delle stesse dimensioni del messaggio originale (che dovrebbero poi essere ridotti).
  - Sono disponibili moltissime librerie di funzioni per l'hashing crittografico.

# Message Authentication Codes (MAC)

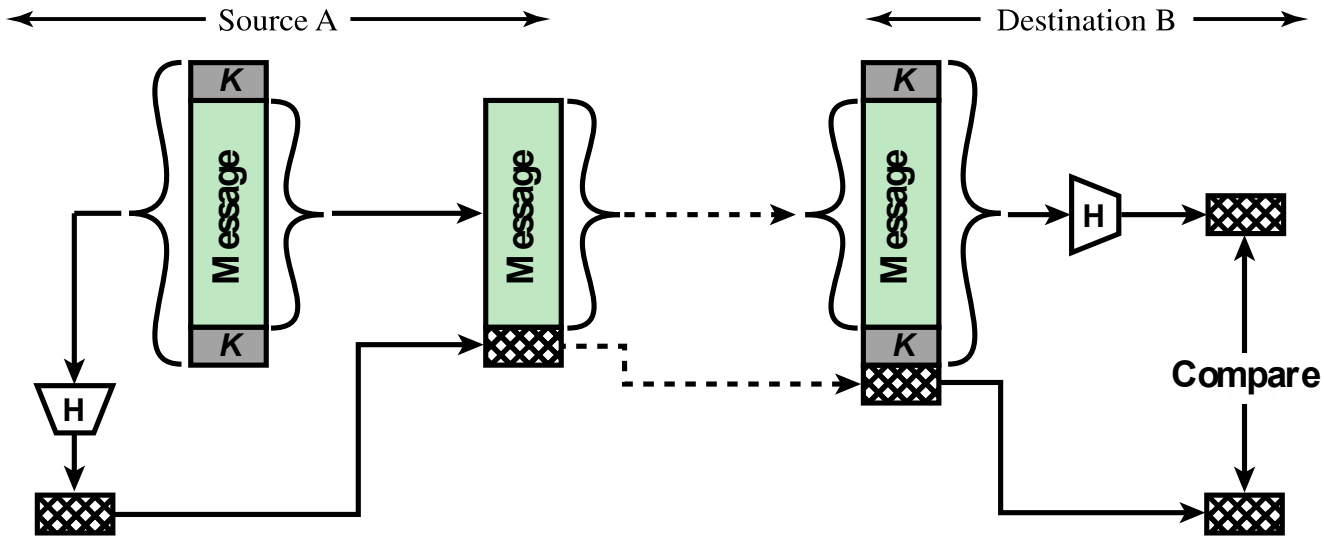



- Una funzione MAC può essere definita come

  MAC = C(K, M)

  dove:
  - M è il messaggio di input
  - C è la funzione MAC
  - K è la chiave segreta condivisa
- MAC è il valore di controllo in output che viene inviato con M.

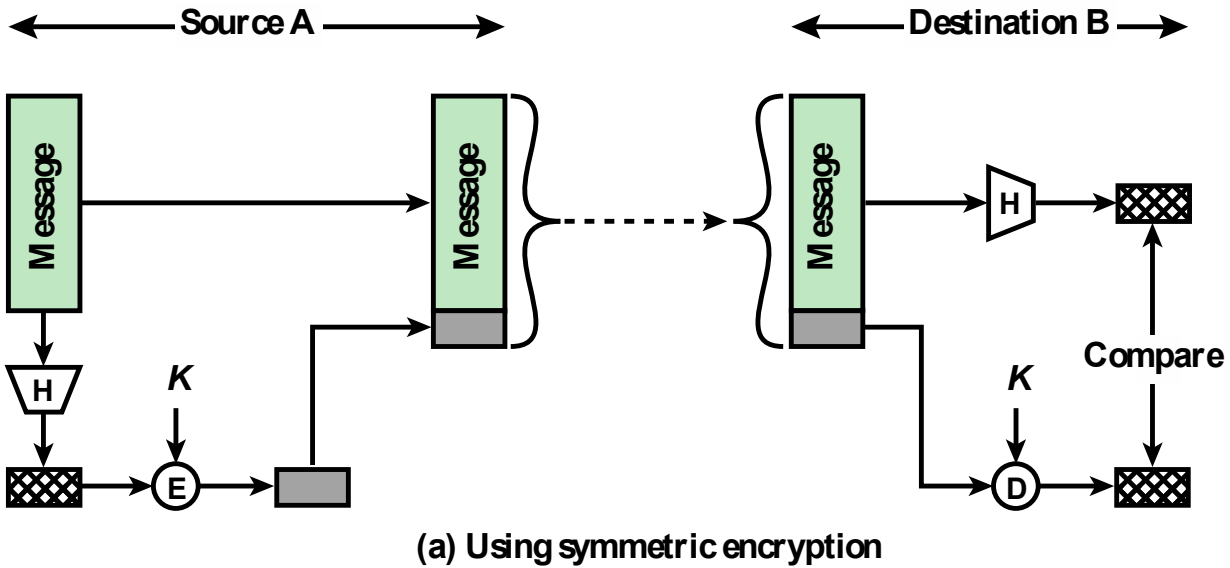
# MAC con hashing e senza crittografia




La chiave segreta condivisa K è incorporata nel processo di generazione di un codice hash.

Picture from: *W. Stalling: Cryptography and Network Security, International Edition, Pearson*

# MAC con hashing e crittografia simmetrica

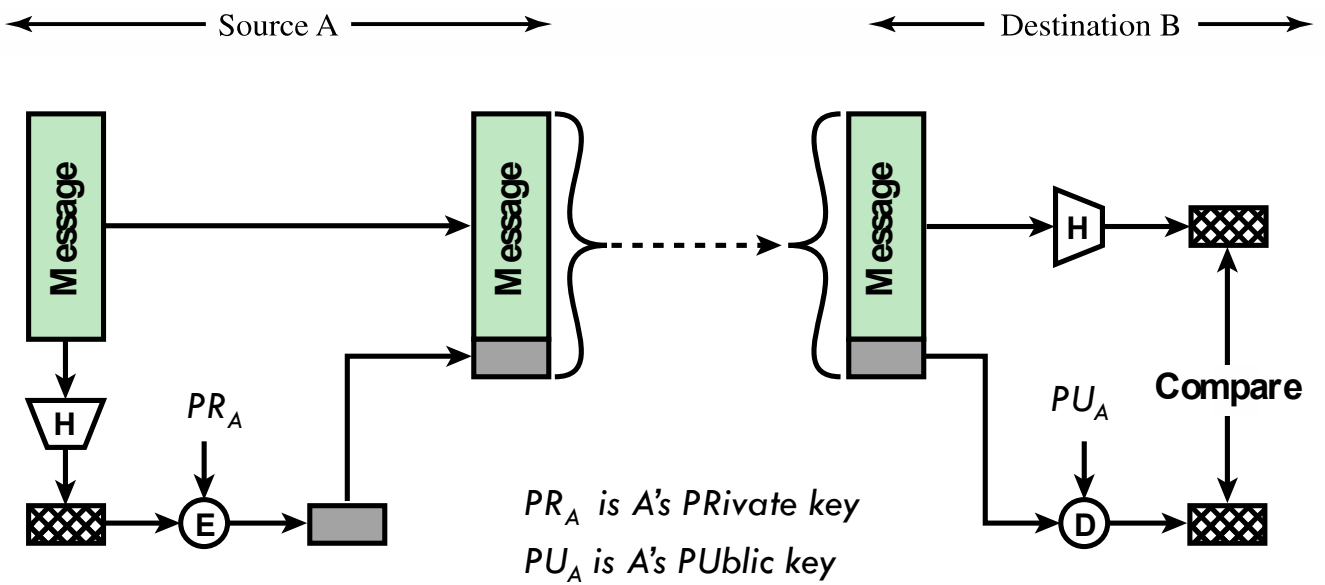



(a) Using symmetric encryption

Picture from: *W. Stalling: Cryptography and Network Security, International Edition, Pearson*

Se solo il mittente e il destinatario condividono la chiave di cifratura, l'autenticità è assicurata.

# MAC con hashing e crittografia asimmetrica




Picture from: *W. Stalling: Cryptography and Network Security, International Edition, Pearson*

La crittografia a chiave pubblica ha il vantaggio di non richiedere la distribuzione delle chiavi.

# Attacchi al MAC

Un MAC può essere soggetto a diversi tipi di attacchi da parte di attaccanti esterni che non hanno la chiave:

- Existential Forgery Attack: L'attaccante è in grado di creare un messaggio M e un MAC validi per M senza avere la chiave. L'attaccante definisce sia M che il MAC corrispondente.
- Selective Forgery Attack: Dato un messaggio M, non scelto dall'attaccante, questo è in grado di produrre un MAC valido per M.
- Universal Forgery Attack: L'attaccante è in grado di creare un MAC valido per ogni possibile messaggio M. Tale attacco è il più potente e implica la totale mancanza di sicurezza dello schema MAC.

# HMAC

- HMAC (Hash-based MAC) è un tipo specifico di MAC che si basa sull'uso di una funzione hash crittografica e di una chiave crittografica segreta.
- HMAC  può essere utilizzato per verificare contemporaneamente l'integrità dei dati e l'autenticità del messaggio.
- Qualsiasi funzione hash crittografica, come SHA-1 o SHA-2, può essere utilizzata nel calcolo di un HMAC; l'algoritmo MAC risultante è chiamato HMAC-X, dove X è la funzione hash utilizzata.

# HMAC: requisiti

- L'HMAC garantisce almeno i seguenti requisiti:
  - Utilizzare funzioni hash disponibili senza doverle modificare.
  - Utilizzare funzioni con buone prestazioni e codice open source.
  - Permettere una rapida sostituzione di una funzione hash con una eventualmente più veloce o più sicura.
  - Utilizzare e gestire le chiavi condivise in modo semplice.
  - Essere resistente ad attacchi MAC.

# Outline

- Funzioni Hash
- Funzioni Hash Crittografiche
- Autenticazione dei messaggi
- Firma digitale e altre applicazioni

# Utilizzo degli hash

- Oltre che per l'autenticazione dei messaggi, l'hash può essere utilizzato per una serie di altri motivi:
  - Password a senso unico (one-way)
  - Rilevamento delle intrusioni
  - Generatore di numeri casuali (PRNG)
  - Firma digitale

# Password one-way

- In alcuni sistemi operativi le password non sono memorizzate per intero, ma solo il loro hash viene conservato nel file delle password.
- Chi entrasse in possesso del file di password non sarebbe in grado di ottenere la password dall'hash.
- Quando un utente digita una password, il sistema calcola l'hash e lo confronta con il valore corrispondente nel file di password.
- Quando si resetta la password e non si riceve in cambio la password in chiaro, spesso è perché il sistema non l’ha memorizzata in chiaro.

# Controllo delle differenze ottimizzato

- Intrusion Detection: Gli hash sono calcolati per ogni file di un computer; se un virus modifica il file, la modifica può essere rilevata ricalcolando l'hash e confrontandolo con quello originale.
- Version control: Molte applicazioni permettono a diversi utenti di modificare contemporaneamente e sincronizzare la modifica sui file condivisi. I servizi di versioning utilizzano funzioni hash per determinare se i file sul server e i file locali sono gli stessi e trasferiscono solo quelli che sono stati modificati.

# Generazione di numeri pseudo casuali

- L'hashing viene utilizzato anche per generare numeri casuali:

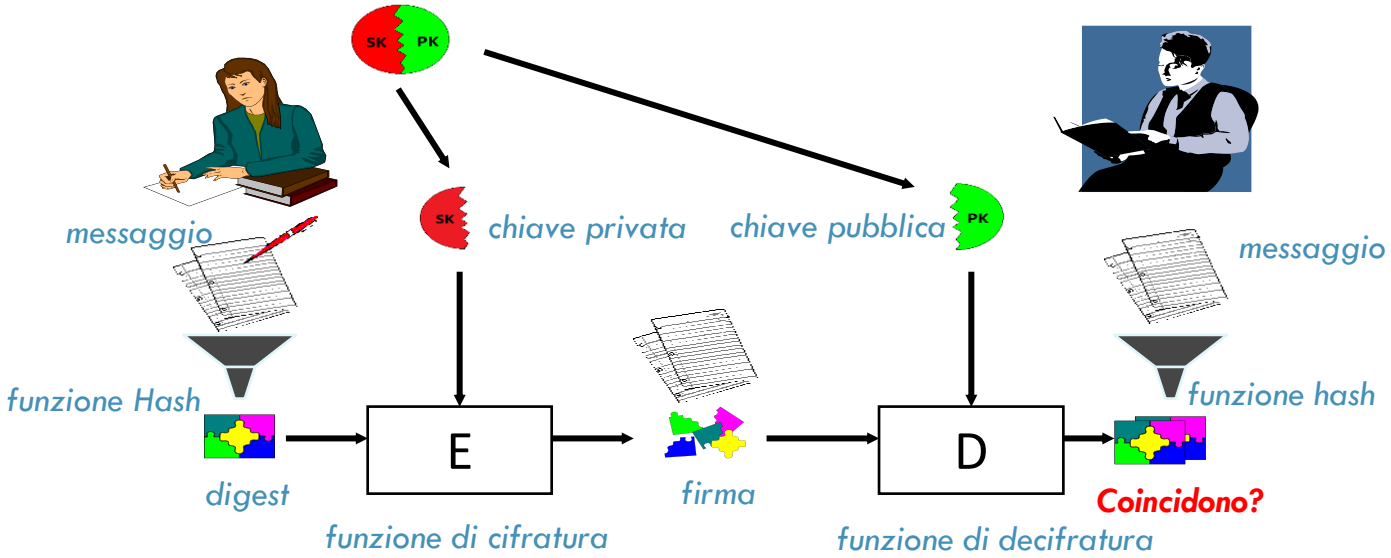
  n = hash(seme).

- Il risultato è un numero effettivamente pseudo-casuale perché una funzione crittografica di hash prende in ingresso una quantità variabile di dati ed emette un blocco a lunghezza fissa.
- Se il seme viene scelto con cura e dato in ingresso ad un hash crittografico, a causa dell'effetto valanga, l'uscita conterrà bit uniformemente distribuiti.

# Firma digitale

- La firma digitale è uno strumento per verificare l'autenticità di messaggi o documenti digitali.
- Una firma digitale valida dà al destinatario un motivo molto forte per credere che il messaggio sia stato creato da un mittente conosciuto (autenticazione) e che il messaggio non sia stato alterato durante il trasporto (integrità).
- L'idea di base è simile al MAC, ma il valore dell'hash è criptato con la chiave privata del mittente.
- Chiunque abbia la chiave pubblica del mittente può verificare l'integrità del messaggio!

# Hashing e firma digitale



# Approfondimento 1: Algoritmo dell'HMAC

L'algoritmo HMAC è un insieme di passi che permette di aggiungere una chiave ad una funzione hash crittografica di base (che è riferita nel riquadro grigio con scritto «Hash», in figura). Tale algoritmo rispetta tutti i requisiti precedenti, ed è riassunto nella figura a lato.

Le componenti principali sono:

H = la funzione hash utilizzata (possono essere utilizzate diverse funzioni hash, come le diverse SHA1 e SHA2)

IV = un valore iniziale in input alla funzione hash

M = il messaggio in input all'HMAC

$Y_i$ = l'i-esimo blocco di M

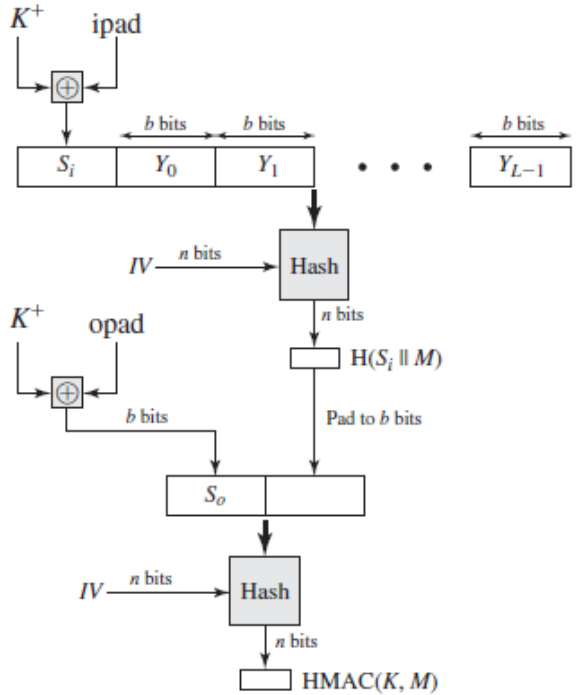
L = numero di blocchi di M

*b* = numero di bit in un blocco

*n* = la lunghezza dell'hash prodotto dalla funzione H

K = la chiave segreta, la cui dimensione raccomandata è >= *n*

$K^+$ = K estesa con degli zero per avere dimensione uguale a b.



# Approfondimento 1: Algoritmo dell'HMAC /2

La funzione HMAC può essere definita come segue:

$$\text{HMAC}(K, M) = \text{H}[(K^+ \oplus \text{opad}) \| \text{H}[(K^+ \oplus \text{ipad}) \| M]]$$

L'algoritmo può essere definito come segue:

Aggiunta di 0 a K per ottenere $K^+$;

XOR tra K+ e *ipad* per ottenere $S_i$;

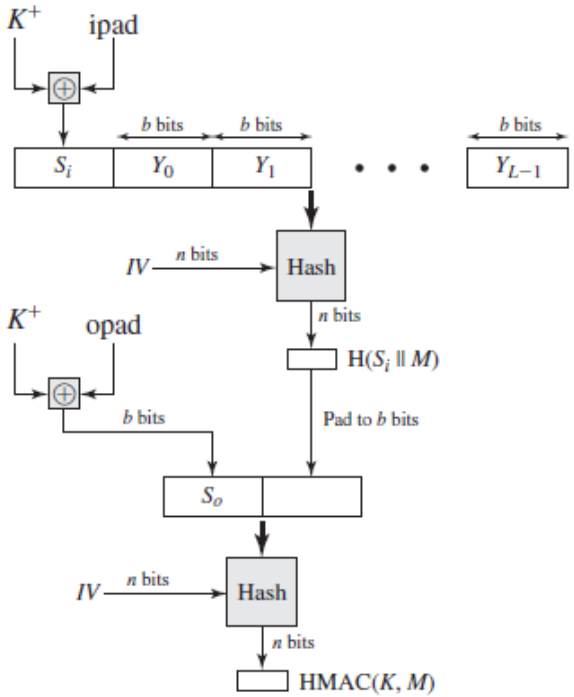
Unire M a $S_i$;

Applicare H allo stream ottenuto al passo 3;

XOR di $K^+$ con *opad* per produrre il blocco $S_o$;

Unire il risultato dell'hash al passo 4 con $S_o$;

Applicare H allo stream generato al passo 6 per ottenere il risultato finale.

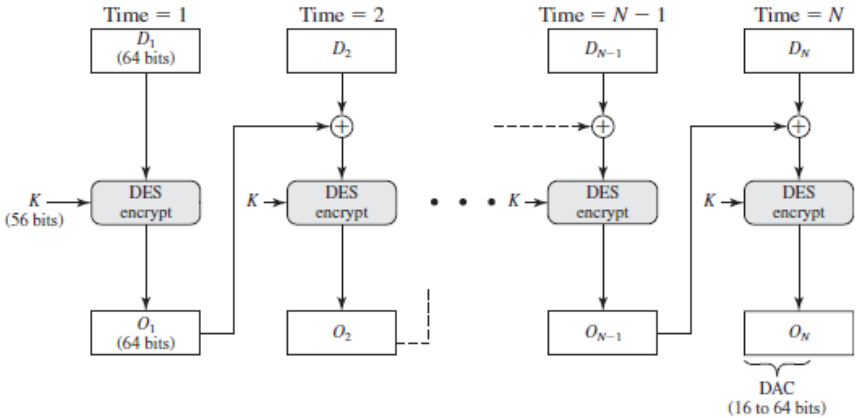


# Approfondimento 2: DAA e CMAC

Esistono *funzioni MAC basate su cifrari a blocchi, anziché su hash, come il* DAA ed il CMAC (che è l'evoluzione del DAA).

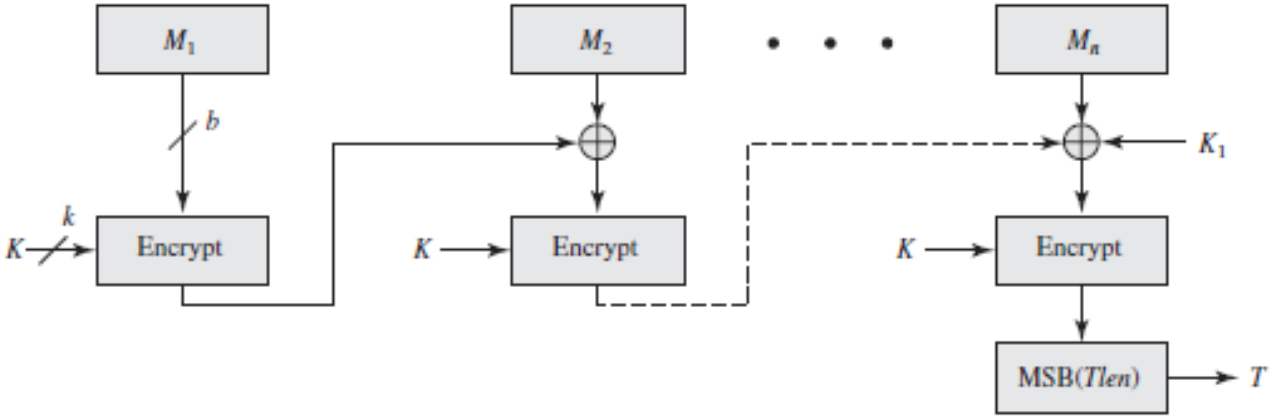
Il **Data Authentication Algorithm** (**DAA**) è basato sul DES ed è stato uno degli algoritmi più usati per anni.

Il DAA è definito come una modalità *Cipher Block Chaining* (ricordate cosa è?) del DES. Il messaggio è diviso in blocchi $D_1$, …, $D_n$ di 64 bit e le operazioni eseguite sono riportate in figura.



# Approfondimento 2: DAA e CMAC

- Il DAA, per certi tipi di messaggi è vulnerabile (a causa di limiti del DES). Per questo motivo è stato proposto un suo raffinamento che supporta sia AES che Triple-DES. La figura sottostante schematizza il comportamento del CMAC.



# Approfondimento 3: Attacchi alle proprietà delle funzioni hash

- Come detto, una funzione hash crittografica deve garantire tre proprietà, ovvero la funzione one-way, la resistenza debole e la resistenza forte alle collisioni.
- Quanto è complesso attaccare queste proprietà in modo da violarle? Più specificamente, quante operazioni di hash sono richieste per violarle?
- Data una funzione hash che restituisce un valore (digest) di dimensione k bit:
  - Per invertire la funzione (pertanto violando la proprietà di one-way) od attaccare la resistenza debole alla collisione, sono richiesti un massimo di $2^k$ operazioni.
  - Per attaccare la proprietà di resistenza forte alle collisioni, sono richiesti un massimo di $2^{k/2}$ operazioni.
- Pertanto, per SHA-1, che restituisce un digest di dimensione 160, sono sufficienti un massimo di $2^{80}$ operazioni per violare la proprietà di resistenza forte alle collisioni. Tale numero di operazioni è basso e sancisce l'inadeguatezza dell'attuale schema SHA-1 per le funzioni MAC moderne.
- All'atto pratico, SHA-1 è stato attaccato con successo con un numero minore si $2^{80}$ operazioni di hash, come vediamo nell'approfondimento successivo.

# Approfondimento 4: Attacchi a SHA-1

Fonte: CERTNAZIONALE.it (https://www.certnazionale.it/news/2017/02/24/shattered-realizzato-il-primo-attacco-reale-di-collisione-sha-1/)

In crittografia, **SHA-1** è una funzione di *hash* che, a partire da un oggetto informatico, ad esempio un file binario o un documento elettronico, produce un *digest* di 160 bit, ossia 20 byte, rappresentati di norma come una stringa esadecimale di 40 caratteri. Questa stringa è teoricamente univoca e viene usata come "impronta" (*fingerprint*) dell'oggetto originale. Se due oggetti differenti generano la stessa impronta SHA-1 si parla di "collisione".

SHA-1 è utilizzato per la firma digitale, nella protezione di comunicazioni elettroniche, per l'identificazione e la verifica di integrità di file in sistemi di archiviazione e piattaforme di controllo versione per lo sviluppo e la distribuzione di software.

Anche se ancora largamente usato, SHA-1 viene oggi considerato non più sicuro e il suo utilizzo non è più consigliato in applicazioni pratiche. Già dal 2010 molte organizzazioni accademiche e di standardizzazione hanno cominciato a considerare SHA-1 "deprecato" e a raccomandare l'uso di altre funzioni di *hash* notevolmente più robuste, come SHA-256 o quelle della famiglia SHA-3. A partire dal 2017, Microsoft, Google, Apple e Mozilla hanno annunciato l'abbandono del supporto a certificati SSL basati su SHA-1 nei loro *browser*.

Attacchi all'algoritmo SHA-1 erano stati già teorizzati da tempo. Nel 2005 sono state dimostrate collisioni dell'algoritmo SHA-1 con uno sforzo computazionale pari a 2^69 operazioni di *hash* (un attacco a forza bruta richiederebbe un minimo di 2^80 operazioni). Più di recente, nell'ottobre del 2015, ricercatori provenienti dal CWI (*Centrum Wiskunde & Informatica*, Paesi Bassi), da Inria (Francia) e NTU (Singapore) hanno reso pubblico quello che viene definito un "freestart collision attack" (noto come The SHAppening) alla funzione di compressione di SHA-1 che richiede "solamente" 2^57 operazioni.

I risultati di queste ricerche, pur minando di fatto la presunta sicurezza di SHA-1, non risultano utilizzabili in pratica per realizzare attacchi reali di collisione, in quanto, al di là dello sforzo computazionale (ed economico) richiesto, un ipotetico attaccante dovrebbe poter agire senza controllo sullo stato interno iniziale dell'algoritmo e, soprattutto, dovrebbe poter scegliere un oggetto arbitrario che produce lo stesso *hash* di un altro oggetto bersaglio.

# Approfondimento 4: Attacchi ad SHA-1

- Ora anche questo scoglio sembra essere stato superato. Il 23 febbraio 2017, ricercatori di **Google** e del gruppo di crittologia del **CWI** olandese hanno annunciato di aver effettuato con successo il primo attacco reale di collisione all'algoritmo SHA-1, battezzato **SHAttered**. In pratica, è possibile predisporre due documenti PDF aventi contenuto differente e generare una firma digitale basata su impronta SHA-1 che risulta valida per entrambi i documenti.
- L'attacco ha richiesto circa 2^63 computazioni di SHA-1 e una potenza di calcolo equivalente a 6.500 anni di calcolo di una singola CPU e 110 anni di una singola GPU. Questo sforzo è stato possibile grazie all'utilizzo di otto *cluster* di CPU dislocati in otto diversi centri di calcolo e di un *cluster* di diverse GPU, tutti ospitati da Google.
- Tutte le applicazioni che si basano su SHA-1 per firme digitali, controllo di integrità o identificazione di file sono potenzialmente vulnerabili a questo tipo di attacco. Google ha già introdotto soluzioni di mitigazione nei suoi servizi Gmail e Google Drive.
- Al momento si esclude che questo attacco possa essere già stato sfruttato da attori malevoli, visto anche il costo notevole che una simile operazione comporterebbe, stimato dagli esperti attorno ai 560.000$ solo per l'affitto delle risorse di calcolo necessarie. Gli autori della ricerca affermano che renderanno disponibile pubblicamente il codice proof-of-concept dell'attacco dopo 90 giorni dall'annuncio.
- Sul sito SHAttered è stato messo a disposizione un *tool* online che consente di verificare la possibilità di una collisione SHA-1 su un singolo documento.

# Approfondimento 5: Proprietà garantite dalle funzioni hash, MAC e firme digitali

| Cryptographic primitive<br>Security Goal | Hash | MAC | Digital signature |
|---|---|---|---|
| Integrity | Yes | Yes | Yes |
| Authentication | No | Yes | Yes |
| Non-repudiation | No | No | Yes |
| Kind of keys | none | symmetric keys | asymmetric keys |

Le firme digitali saranno trattate nel modulo 2.10