



Corso di Crittografia

Prof. Dario Catalano

Message Authentication



Introduzione

- Fino ad ora abbiamo parlato di sicurezza in termini di privacy.
- Garantire autenticita' e' un obiettivo forse ancora piu' importante.
 - Non sempre la privacy e' una necessita'
 - Quasi sempre e' necessario essere certi circa l'identita' del mittente.
- Oggi studieremo metodi (simmetrici) per realizzare tale obiettivo.



Obiettivi

- Il mittente S vuole inviare un msg M al destinatario R, in modo che se M e' corrotto, R possa rendersene conto.
- Viene garantita l'autenticita' del mittente.
- Viene protetta l'integrita' del messaggio.



Il nostro contesto

- S ed R hanno una chiave segreta k in comune.
- L'autentica è per il "bene" di R, ma è prodotta da S (utilizzando k e un algoritmo appropriato).
- L'autentica di un messaggio M produce un messaggio M' .
- Se R riceve M'' , dovrà essere in grado di:
 - Ricavare il messaggio originale M
 - Convincersi che M'' non è autentico.



Message Authentication Codes

- Spesso M' e' proprio il messaggio originale M , con aggiunta una stringa (di lunghezza fissa) *tag*.
- Il tag serve a R per verificare l'autenticita' di M .
- In tali casi si parla di Message Authentication Codes (MAC).
- Se il messaggio non risulta autentico il comportamento di R dipende dal contesto.



L'avversario

- L'avversario deve partecipare ad una comunicazione.
 - Per sperare di avere informazioni, qualcosa deve essere inviata a R.
- Supporremo che A controlla la rete di comunicazione.
- Obiettivo: capire come (e se) partecipare in modo attivo alla conversazione, aiuta l'avversario.



Autenticita' via Privacy

- Cifrare puo' garantire autenticita'?
- Se S vuole mandare M a R, S calcola $M' = \text{Enc}(M)$ e invia M' a R.
- R decifra, se il risultato ha senso allora esso viene dichiarato autentico.
- Ragionamento intuitivo: A non conosce M → non e' in grado di modificarlo in modo corretto.



Eppure...

- Potremmo essere in grado di modificare il messaggio senza necessariamente conoscerlo (parzialmente o interamente).
- Che succede se utilizzassimo cifrari piu' potenti (o semplicemente diversi)?
 - Es. cifrari sicuri contro attacchi attivi.



Autenticita' vs Privacy

- Il problema non sta' nel cifrario utilizzato.
- Piu' semplicemente gli obiettivi sono diversi.
- Utilizzare per un dato problema, metodi progettati per risolverne altri e' sbagliato.
 - Sappiamo che obiettivi vogliamo realizzare, non come (o da che contesto) nascono tali obiettivi.
 - Le nostre definizioni sono studiate per obiettivi precisi e specifici.



MAC - Definizione

(KeyGen, MAC, VF)

- L'algoritmo KeyGen restituisce una stringa casuale (nell'insieme opportuno)
- MAC. Prende in input una chiave k , un messaggio $M \in \{0, 1, \}^*$. Restituisce $tag \in \{0, 1, \}^* \cup \{\perp\}$.
- VF (deterministico). Prende in input una chiave k , un messaggio $M \in \{0, 1, \}^*$ e un $tag \in \{0, 1, \}^*$. Restituisce un bit d



Commenti

- Fino ad ora non abbiamo parlato di sicurezza.
- La sintassi descritta sembra imporre verificatori (destinatari) deterministici.
- In realta' avere destinatari a stati non e' impossibile ma problematico.
 - L'avversario controlla la comunicazione → non e' banale prevedere lo stato del destinatario.
- Eppure algoritmi di verifica a stati sono importanti per far fronte a certi tipi di attacco (es. replay attacks)



Sistemi Deterministici

- Per il caso dei cifrari abbiamo visto che I metodi deterministici non possono essere sicuri.
- Che possiamo dire per l'autentica dei messaggi?
- Vale la stessa restrizione?



Sistemi Deterministici (cont.)

- Se MAC e' deterministico (e senza stati)
l'algoritmo di verifica e' sempre lo stesso

VF_k(*M*, *tag*)

tag' ← *MAC*_k(*M*)

if *tag*' = *tag* ∧ *tag* ≠ ⊥ **return** 1 **else return** 0.

- Dunque per tali MAC non abbiamo bisogno di specificare un algoritmo di verifica.

Verso una definizione di sicurezza



- Obiettivo: rivelare ogni tentativo di modifica da parte di A.
- Il destinatario dovrebbe considerare autentico solo un msg M che non e' stato alterato.
- Se A produce (M, tag) tali che $\text{VF}_k(M, \text{tag})=1$, diciamo che A produce un *falso*.
- Come possiamo formalizzare questo requisito?



Requisiti

- Determinate proprietà sono certamente necessarie (es. segretezza della chiave)
- L'obiettivo però rimane la non falsificabilità.
- Determinati "falsi" potrebbero non avere alcun significato.
- E' giusto considerarli veri e propri falsi?

LA RISPOSTA È SÌ, PERCHÉ VF
TORNERÀ 1



Attacchi

- Una volta stabiliti gli obiettivi restano da definire i tipi di attacchi che fronteggeremo.
- Dobbiamo stabilire a cosa ha accesso l'avversario prima di (tentare di) produrre un falso.
- Possiamo definire diverse classi di avversari via via piu' potenti.



No message attack

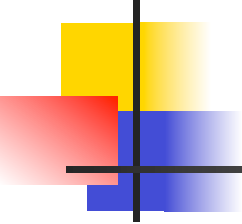
- A non ha la possibilità di accedere a nulla di più che la descrizione degli algoritmi utilizzati.
- A deve produrre una coppia valida (M, tag) .
- Si tratta di un attacco non molto realistico.



Replay Attack

- A intercetta una coppia (valida) (M, tag) e la invia una seconda volta a R.
- Se il MAC è deterministico e senza stati tale attacco è inevitabile: cioè che R accetta la prima volta, verrà accettato ogni altra volta.
- Dobbiamo considerare questo un attacco importante?

SÌ, MA NEL NOSTRO CASO NON
VERRÀ PRESO IN CONSIDERAZIONE

- 
-
- In realta' non vi e' motivo perche' A debba vedere una sola coppia (M, tag)
 - Potrebbe avere a disposizione q_s coppie di tale tipo.
 - Tale numero puo' essere limitato in contesti specifici.
 - In generale, tratteremo q_s come una importante risorsa di A.



Attacchi a msg scelto (CMA)

- A potrebbe vedere q_s coppie sulle quali non ha alcun controllo.
- Alternativamente, potremmo immaginare di fornire ad A un oracolo che restituisce il tag opportuno ogni qualvolta A chiede un msg M.
- In tal caso parliamo di attacco a messaggio scelto (CMA).
- L'obiettivo diventa di produrre un falso MAC per un messaggio non richiesto all'oracolo.



Completiamo il modello

- A potrebbe essere in grado di produrre diversi candidati "falsi".
- A vince se anche solo uno di tali candidati e' valido.
- Possiamo formalizzare tale capacita' permettendo ad A di accedere ad un oracolo di verifica.



Ricapitoliamo

- A può chiedere di autenticare un certo numero di messaggi.
 - A ha a disposizione q_s richieste di autentica.
- A può chiedere se una coppia del tipo (M, tag) è valida
 - A ha a disposizione q_v richieste di verifica.
- A vince se effettua una richiesta di verifica, su un messaggio per il quale non aveva ricevuto autentica, e l'algoritmo di verifica restituisce 1.



Definizione (PAG. 9)

- MA: (KeyGen, MAC, VF)

$\text{Esp}_{\text{MA}}^{\text{uf-cma}} (A)$

$k \leftarrow_R \text{KeyGen}();$ Esegui $A^{\text{MAC}_k(\cdot), \text{VF}_k(\cdot, \cdot)}$

if A fa una richiesta di verifica (M, tag) tale che

- L'oracolo di verifica risponde 1
- A non aveva richiesto M all'oracolo MAC

Return 1

else Return 0

$$\text{Adv}^{\text{uf-cma}} (A) = \Pr[\text{Esp}_{\text{MA}}^{\text{uf-cma}} (A) = 1]$$



Commenti

- E' importante separare q_s e q_v dal tempo di calcolo.
- In pratica le autentiche possono essere ottenute solo dal legittimo mittente.
- q_s e q_v possono essere limitate dal sistema.
- Adesso facciamo pratica con la definizione guardando un esempio.



Esempio

- $F: \{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^L$ PRF

MAC_k(*M*)

if ($|M| \bmod \ell \neq 0 \vee |M| = 0$) **return** \perp

Sia $M = M[1] \cdots M[n]$ // $|M[i]| = \ell$

for $i = 1, \dots, n$ $y_i \leftarrow F_k(M[i])$

$Tag \leftarrow y_1 \oplus \cdots \oplus y_n$

return *Tag*

- E' questo schema sicuro?

POSSIAMO EFFETTUARE UN NO MSG
ATTACK



Il paradigma PRF-come-MAC

- PRF consentono di costruire MAC sicuri
- In realta' tale approccio e' ottimo.
- La sicurezza del MAC costruito nel modo che vedremo e' praticamente la stessa della PRF utilizzata (la dimostrazione non degrada di molto la sicurezza)



MAC da PRF

- $F: K \times D \rightarrow \{0,1\}^\tau$ fam. di funz.

KeyGen

$k \leftarrow_R K$

return k

$\text{MAC}_k(M)$

If $(M \notin D)$ return \perp

Tag $\leftarrow F_k(M)$

return Tag

- Non specifichiamo l'algoritmo di verifica poiche' il metodo e' deterministico.



Teorema

- $F: K \times D \rightarrow \{0,1\}^\tau$ fam. di funz.
- $MA=(\text{KeyGen},\text{MAC})$ il metodo appena descritto.
- A avversario che attacca MA facendo $(q_s + q_v)$ domande.
- Esiste un avversario B - contro F - che fa $(q_s + q_v)$ domande e tale che

$$Adv_{MA}^{uf-cma}(A) \leq Adv_F^{prf}(B) + \frac{q_v}{2^\tau}$$

DIMOSTRAZIONE

Bk:

$d \leftarrow 0$; $S \leftarrow \emptyset$

RUN A

WHEN A ASKS ITS SIGNING ORACLE SOME QUERY M :

ANSWER $f(M)$ to A; $S \leftarrow S \cup \{M\}$

WHEN A ASKS ITS VERIFICATION ORACLE (M, tag) :

if $f(M) = TAG$ then

ANSWER 1 to A;

if $M \notin S$ then $d \leftarrow 1$

else ANSWER 0 to A

UNTIL A halts

return d

$\Pr[\text{ESP}_F^{\text{prf}-1}(B) = 1] = \text{Adv}_{\text{RA}}^{\text{uf-cma}}(A)$
PERCHÉ DIPENDE DALLA BRAVURA DI A

$$\Pr[\text{ESP}_F^{\text{prf}-0}(B) = 1] = \frac{q_N}{2^\tau}$$

$\frac{1}{2^\tau}$ È LA PROBABILITÀ CHE $\text{Tag} = f(m)$

SICCOME VENGONO FATTE q_N RICHIESTE

SI HA $q_N/2^\tau$



CBC-MAC

- Si tratta di un metodo molto diffuso per costruire MAC a partire da un cifrario a blocchi $E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
- Metodo deterministico e senza stati.
- KeyGen si limita a restituire una stringa casuale di k bit.



MAC_k(M)

if $M \notin \mathcal{M}$ return \perp

// \mathcal{M} spazio dei messaggi

Sia $M = M[1] \cdots M[m]$

// $|M[i]| = n$

$C[0] \leftarrow 0^n$

for $i = 1, \dots, m$

$C[i] \leftarrow E_k(C[i - 1] \oplus M[i])$

return $C[m]$ *// Tag = C[m]*



Teorema

- $E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ cifrario a blocchi
- $\{0,1\}^{nm}$ spazio dei messaggi
- A avversario contro CBC-MAC che fa q richieste di autentica e 1 richiesta di verifica.
- Esiste B che attacca E come PRP, che fa $q+1$ domande e tale che

$$\text{Adv}_{MAC}^{uf-cma}(A) \leq \text{Adv}_E^{prp-cpa}(B) + \frac{m^2 q^2}{2^{n-1}}$$



Osservazioni

- Il teorema assume che tutti i messaggi abbiano la stessa taglia.
- Questa e' un'ipotesi necessaria per la validita' del teorema.
- Si puo' facilmente verificare che se si considerano messaggi di lunghezza arbitraria CBC-MAC non e' piu' sicuro.



Contro esempio

- Si consideri il seguente avversario che fa solo due domande e rompe CBC-MAC
 - $M_1 = 0^n$ (msg di un solo blocco)
 - $M_2 = M_2[1] || \dots || M_2[l]$ (msg arbitrario di l blocchi)
- Siano tag_1 e tag_2 le risposte ricevute
- A restituisce $(M_3, tag_3) = (M_2 || tag_2, tag_1)$



Ulteriori osservazioni

- A differenza di CBC\$, CBC-MAC è deterministico.
- Questa proprietà è **cruciale** per la sicurezza di CBC-MAC

IDEA: Se (R, tag) MAC di $M[1] \dots M[m]$, calcolo (R', tag) MAC di $M'[1] \dots M[m]$

$$R \oplus M[1] = R' \oplus M'[1]$$



CBC-MAC per messaggi di lunghezza variabile

Primo approccio: utilizzare chiavi (computazionalmente) indipendenti per messaggi di lunghezza diversa.

F pseudorandom function, ℓ lunghezza del messaggio m .

1. $k_\ell = F_k(\ell)$
2. $\text{tag} = \text{CBC-MAC}(k_\ell, m)$



CBC-MAC per messaggi di lunghezza variabile

Secondo approccio: “prependere” a m la sua lunghezza (stringa di n bit).

Terzo approccio: due chiavi indipendenti k_1, k_2

1. $t = \text{CBC-MAC}(k_1, m)$

2. $\text{tag} = \text{CBC-MAC}(k_2, t)$

- Pro: non è necessario conoscere $|m|$ in anticipo
- Contro: 2 chiavi (ma si può ovviare usando prf)



MAC from hash functions

- Abbiamo visto come costruire MAC da cifrari a blocchi
- Possiamo ottenere MAC anche da funzioni hash costruite secondo la metodologia Markle-Damgard

Pro: La maggior parte delle funzioni hash utilizzate in pratica segue tale metodologia

Contro: Dobbiamo assumere che la funzione di compressione abbia proprietà di pseudocasualità



Merkle-Damgard

H(K, M)

$y \leftarrow \text{pad}(M)$

Sia $y = M_1 || M_2 || \dots || M_n$
// ($|M_i| = b$)

$V \leftarrow \text{IV}$

// IV vettore iniziale di v -bit

for $i = 1, \dots, n$

$V \leftarrow h(K, M_i || V)$

Return V



HMAC - Preliminari

Facciamo le seguenti ipotesi

- H, h restituiscano output di n bit.
- Se m è nell'insieme $\{0,1\}^b$, $H(m)$ richiede una sola esecuzione di h .
- $IV, opad, ipad$ costanti di lunghezza b .
 - $opad$ contiene il byte $0x36$ (ripetuto quanto serve)
 - $ipad$ contiene il byte $0x5C$ (come sopra)



HMAC

KeyGen \rightarrow POTREBBE NON SERVIRE PER
FUNZIONI HASH KEYLESS

- $s \leftarrow \text{HASH-KeyGen}.$
- Sceglie una stringa casuale k di b bit.
 - b lunghezza del blocco

$\text{MAC}((s,k),m) \quad |m|=L$

$$\text{tag} \leftarrow H_s((k \oplus \text{opad}) || H_s(k \oplus \text{ipad} || m))$$



HMAC in pratica

- Molto efficiente e facile da implementare
- Standard industriale, molto utilizzato in pratica.
- Nasce come alternativa a CBC-MAC considerato troppo lento
 - Questo comportava che spesso in pratica si utilizzassero costruzioni del tutto insicure
 - Es: $\text{tag} = H_s(k || m)$ (perché non è sicuro?)

$$\text{pad}(k \| m) = y = k \| m \| 1 \| 0^d \| l$$

$$H(k \| m) = \text{tag}$$

Sia m' UN TESTO ARBITRARIO

$$\hat{m} = m \| 1 \| 0^d \| l \| m'$$

$$H(k, \hat{m}) = \underbrace{k \| m \| 1 \| 0^d \| l \| m'}_{\text{tag}} \| 1 \| 0^{d'} \| l'$$

RICAPITOLANDO

ENCRYPT then MAC

VS

MAC then ENCRYPT