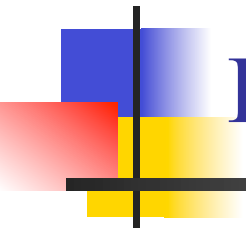
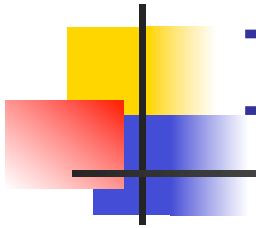


Corso di Crittografia

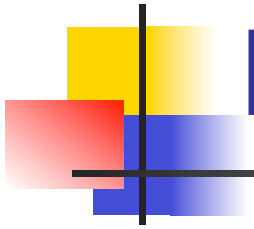
Prof. Dario Catalano

Firme Digitali



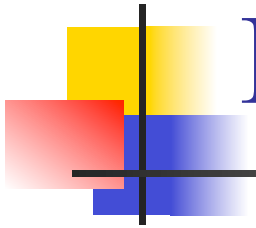
Introduzione

- Una firma digitale e' l'equivalente informatico di una firma convenzionale.
- Molto simile a MA, solo che qui abbiamo una struttura asimmetrica.
  - La chiave segreta e' utilizzata per creare firme
  - La chiave pubblica per verificarle



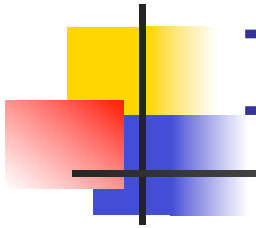
Differenza fondamentale

- Una firma digitale e' (in generale) non ripudiabile.
- La natura asimmetrica di tale primitiva fa si che solo un utente (colui che conosce la chiave segreta) possa firmare.
- Nei MAC non vi e' sostanziale differenza tra colui che autentica e colui che verifica.



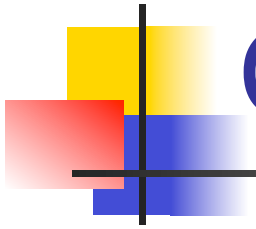
Formalmente

- Uno schema di firma digitale consiste nei seguenti algoritmi
  - Un algoritmo di generazione della chiave KeyGen
  - Un algoritmo di generazione della firma
  - Un algoritmo di verifica



In particolare

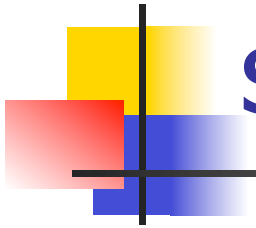
- L'algoritmo KeyGen (randomizzato) restituisce una coppia (vk,sk) appartenente all'insieme Chiavi(k).
- L'algoritmo Sign (randomizzato, deterministico o a stati) prende in input sk e un messaggio m e restituisce una firma σ (o un simbolo speciale $\perp$)
- L'algoritmo Ver (deterministico) prende in input (vk,m,σ) e restituisce un bit.



Considerazioni

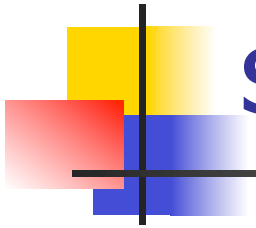
- Nel caso in cui il sistema sia senza stati, bisogna specificare uno spazio di messaggi validi.
- Tale insieme non viene specificato nel caso di schemi di firma a stati, perche' la validita' di un messaggio puo' dipendere dallo stato.
- L'algoritmo Ver restituisce 1 per ogni firma correttamente generata.

Verso una definizione di sicurezza

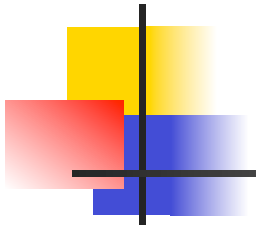


- Gran parte del lavoro lo abbiamo fatto per MA.
- Qui abbiamo una struttura piu' flessibile, dobbiamo solo adattare la definizione vista per MA al nuovo contesto.
- Ripassiamo qualcuna delle considerazioni fatte per MA

Verso una definizione di sicurezza (cont.)



- 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,σ) tali che $Ver(vk,M,\sigma)=1$, diciamo che A produce un *falso*.



- Anche nel nuovo contesto possiamo immaginare di dividere l'attacco in due fasi
  - Dapprima A richiede la firma di un certo numero di messaggi.
  - Quindi prova a costruire un "buon" falso
- Concediamo ad A solo un oracolo di firma.
  - A non ha alcun bisogno di un oracolo di verfica.

Definizione

- DS: (KeyGen,Sign,VF)

$Esp_{DS}^{uf\text{-}cma}$ (A)

(vk,sk) $\leftarrow_R$ KeyGen(); (M,σ) $\leftarrow$ $A^{Sig(sk,.)}$(vk);

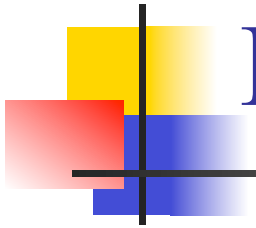
if (Ver(vk,M,σ)==1) & (M $\in$ Messaggi) &

(A non aveva richiesto M all'oracolo)

Return 1

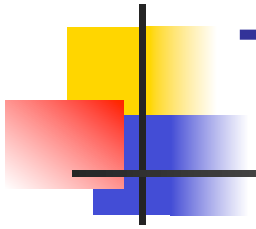
else Return 0

$Adv^{uf\text{-}cma}$ (A)=Pr[$Esp_{DS}^{uf\text{-}cma}$ (A) =1]



Firme RSA

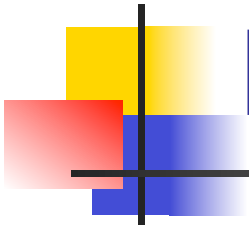
- La funzione RSA e' molto utilizzata anche come primitiva di base per costruire schemi di firme digitali.
- Partiremo da alcuni schemi di base e poi vedremo soluzioni via, via piu' sofisticate.
- In ogni caso, l'algoritmo di generazione delle chiavi per i sistemi basati su RSA e' praticamente sempre lo stesso.



Trapdoor Signatures

Idea: sfruttare la funzione trapdoor in modo inverso a quanto fatto per cifrare.

- Nel caso di RSA:
  - L'algoritmo di cifratura diventa l'algoritmo di verifica.
  - L'algoritmo di decifratura diventa l'algoritmo di firma
  - L'algoritmo di gen delle chiavi resta invariato.



Piu' precisamente

Sign$_{N,d}(M)$

If $M \notin Z_N^*$ return $\perp$; $\sigma \leftarrow M^d \bmod N$

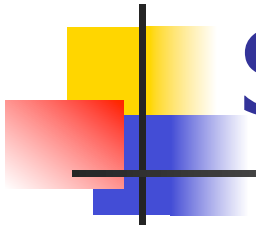
Return σ

Ver$_{N,e}(M, \sigma)$

If $(M \notin Z_N^* \vee \sigma \notin Z_N^*)$ return $\perp$

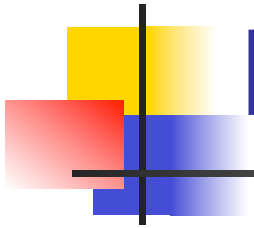
if $\sigma^e = M \bmod N$ Return 1

else Return 0



Sicurezza dello schema

- Sappiamo che invertire RSA e' un problema difficile.
- E' questo sufficiente a garantire un qualche livello di sicurezza al nostro schema?



Hash e Inverti

- Dobbiamo trovare un modo per “distruggere” le proprieta’ algebriche di RSA.
- Inoltre sarebbe meglio poter assumere che M e’ una stringa qualunque e non un elemento di un gruppo.

Idea: modificare lo schema utilizzando una funzione hash

Hash e Inverti (cont)

Sign$_{N,d}(M)$

$y \leftarrow H_N(M)$; // $H_N : \{0,1\}^* \rightarrow Z_N^*$

$\sigma \leftarrow y^e \bmod N$

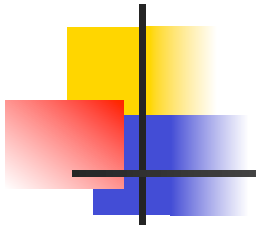
Return σ

Ver$_{N,e}(M, \sigma)$

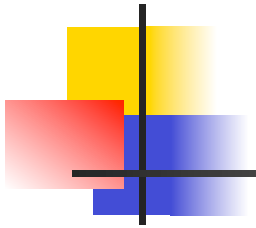
$y \leftarrow H_N(M)$; $y' = \sigma^e \bmod N$;

if $(y = y')$ Return 1

else Return 0

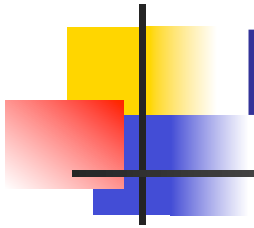


- Questa soluzione risolve certamente il problema dell'arbitrarieta' dei messaggi.
- Intuitivamente sembra risolvere anche il secondo problema.
  - Una buona funzione hash, dovrebbe essere in grado di "distruggere" le proprieta' algebriche che ci davano noia.
- Pero' essa fa sorgere un problema che non avevamo in precedenza.



- In prima istanza, potremmo isolare certe proprieta' necessarie affinche' lo schema possa funzionare.
  - Ad es. Resistenza alle collisioni
- Il nostro obiettivo pero' rimane dimostrare che l'approccio e' sicuro.
  - Limitarsi a condizioni necessarie puo' indurre all'errore.
  - Bisogna stabilire condizioni sufficienti.

Full Domain Hash



- Supponiamo di affrontare il problema in modo diverso
- Che succede se avessimo a disposizione una funzione hash perfetta (RO)?
  - Se hash-e-inverti ha problemi anche in questo caso sarebbe opportuno cambiare approccio.
  - Tale esperimento e' teorico perche' funz. Hash perfette non ne esistono.

Uf-cma di Hash e Inverti (nel ROM)

$Esp_{DS}^{uf\text{-}cma}$ (A)

$((N,e),(p,q,d)) \leftarrow_R KeyGen^{RSA}()$;

$H \leftarrow_R Func(\{0,1\}^*, Z_N^*)$;

$(M,\sigma) \leftarrow A^{H(.),\ Sign^H_{N,d}(.)}\ (N,e)$;

if $(Ver_{N,e}(M,\sigma) = 1)$ & $(M \in$ Messaggi) &

(A non aveva richiesto M all'oracolo)

Return 1

else Return 0

$Adv^{uf\text{-}cma}(A) = Pr[Esp_{DS}^{uf\text{-}cma}(A) = 1]$

Teorema

- Sia DS=($KeyGen_{RSA}$,Sign,Ver), lo schema descritto.
- Sia A un avversario che al piu' q_s richieste di firma e q_H richieste all'oracolo ($q_H > q_s$)
- Esiste un avversario I che usa risorse comparabili ad A, tale che

$$Adv^{uf\text{-}cma}(A) \leq q_H \, Adv^{ow\text{-}rsa}(I)$$