Identificazione, Autenticazione e Firma Digitale

Ozalp Babaoglu

ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

Prologo

- In origine crittografia = confidenzialità
- Diffusione delle reti: nuove funzionalità
  - Identificazione
  - Autenticazione
  - Firma digitale

Prologo

- **Identificazione**: un sistema di elaborazione deve essere in grado di accertare l'identità di un utente che vuole accedere ai suoi servizi
- **Autenticazione**: il destinatario di un messaggio deve essere in grado di accertare l'identità del mittente e l'integrità del messaggio
- **Firma digitale**: funzionalità complessa richiesta quando mittente e destinatario di un messaggio non si fidano l'uno dell'altro (reciprocamente). Proprietà simili alla firma cartacea

Funzioni Hash

x: Pre-image
y: Hash value (also known as *digest* or *fingerprint*)

$$f : X \mapsto Y$$

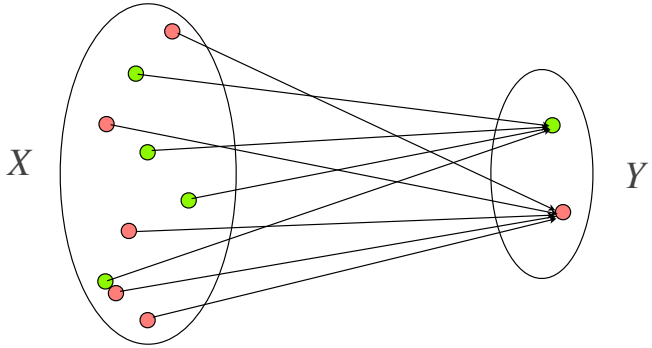
$$|X| = n, \ |Y| = m, \ n \gg m$$

$$y = f(x) \in Y = fingerprint \ di \ x$$

Proprietà di funzioni Hash

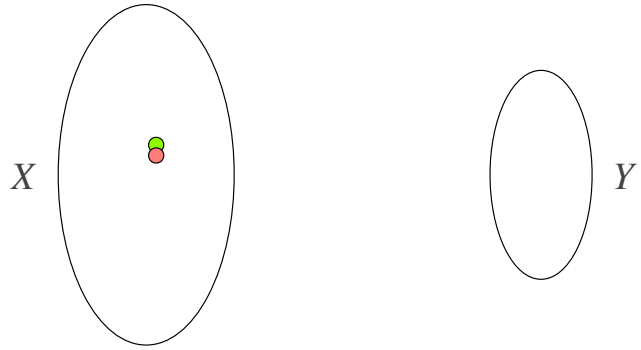
$$X_i = \{x \in X : \ f(x) = y_i\}, \ 1 \leq i \leq m$$

$$|X_1| \cong |X_2| \cong \cdots \cong |X_m|$$



Proprietà di funzioni Hash

Elementi molto simili in X appartengano a sottoinsiemi distinti



Funzioni Hash in crittografia: Hash one-way

1. For any $x \in X$, it is easy to compute $f(x)$
2. For any $y \in Y$, it is computationally infeasible to find $x \in X$ such that $f(x)=y$
3. Given any x_1, it is computationally infeasible to find another x_2 such that $x_2 \neq x_1$ and $f(x_1)= f(x_2)$

Funzioni Hash one-way

Simple example (8-bit block parity):

```
m=110100101000100111100101000101001010001000010 1
```

```
b1=11010010
b2=10001001
b3=11100101
b4=00010100
b5=10100010
b6=00010100
digest=00011100 (column-wise ⊕)
```

Not secure! Does not satisfy the second and third properties

Funzioni Hash one-way

- Esempi pratici: MD2, MD4, MD5, SHA-1 (160-bit hash values), Snefru, HAVAL
- MD5 example: 128-bit digest

```
MD5("The quick brown fox jumps over the lazy dog")
  = 9e107d9d372bb6826bd81d3542a419d6


MD5("The quick brown fox jumps over the lazy eog")
  = ffd93f16876049265fbaef4da268dd0e
```


Firma Digitale

ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

Firma Manuale: Proprietà

- La può generare solo una persona e non è falsificabile
- Non è riutilizzabile (legata al documento su cui è apposta)
- Il documento su cui è apposta non è modificabile
- Non può essere ripudiata da chi l'ha generata

Firma Manuale e Firma Digitale

- La firma digitale deve soddisfare le proprietà di quella manuale
- La firma digitale è una stringa di bit e quindi è facilmente duplicabile/copiabile. La firma manuale no!!
- Firma manuale e firma digitale sono fisicamente oggetti di natura completamente diversa!

Digital Signatures

- **Authentic**: Convinces the recipient that the signer deliberately signed the document
- **Unforgeable**: Proof that the signer, and no one else, deliberately signed the document
- **Not reusable**: Signature part of document and cannot be moved to another document
- **Unalterable**: After the document is signed, it cannot be altered
- **Cannot be repudiated**: The signer cannot claim to *not* have signed the document

Digital Signatures

- **Signing** a message with private key K

  $S_K(M)$
- **Verifying** a signature with the corresponding public key

  $V_K(M)$
- **Authentication**: the receiver of a message is convinced of the *identity* of the sender and the *integrity* of the message

Firma Digitale: Implementazione Ingenua

- Digitalizzazione della firma manuale, ad esempio usando uno scanner
- Attacco: Copia e Incolla!!!

Protocol 0: Symmetric Cryptography with Arbitrator

- A wants to send a signed message to B
- T is a trusted arbitrator sharing secret keys K_A and K_B with A and B
- A sends $C(K_A,m)$ to T
- T decrypts to obtain m from $D(K_A,C(K_A,m))$
- T sends to B

  $C(K_B,<m,$"I certify having received m from A"$>)$
- B decrypts the bundle, reads the message and T's certification that A sent it

Protocol 1: Asymmetric Cryptography

A: Sign

```
s=D(m,kA[priv])   //sign
send <A,m,s>
```

B: Verify

```
m*=C(s,kA[pub])   //verify
if m*=m then yes else no
```


Osservazioni

- Occorre un cifrario commutativo, cioè

$$D(C(m)) = C(D(m)) = m$$

- RSA è commutativo
- Il documento firmato non è indirizzato ad uno specifico destinatario
- Tutti possono fare la verifica

Proprietà del protocollo

- La può generare solo una persona: colui che conosce $k_A[priv]$ cioè A
- Non è riutilizzabile/copiabile/falsificabile: in quanto è funzione del documento su cui è apposta
- Il documento su cui è apposta non è modificabile: in quanto anche la firma andrebbe nuovamente generata
- Non può essere ripudiata da chi l'ha generata: in quanto solo conoscendo $k_A[priv]$ è possibile generarla

Difetti del protocollo

- La lunghezza del messaggio scambiato è il doppio della lunghezza del messaggio originario
- Il messaggio non può essere confidenziale in quanto dall'operazione di verifica è possibile risalire al messaggio spedito

Protocol 2

A: Sign and Encrypt

```
s=D(m,kA[priv])    //sign
c=C(s,kB[pub])     //encrypt
send <A,c>
```

B: Decrypt and Verify

```
s*=D(c,kB[priv])   //decrypt
m*=C(s*,kA[pub])   //verify
if m* "makes sense"
       then yes else no
```


Comments

- Signing before encrypting is prudent and more secure
- The other way around is akin to “blind signing”
- Consider
  - Digital signature = manual signature
  - Encryption = envelope
  - You would not sign a sealed envelope since you don’t know what letter is inside
  - Would have little legal force

In Practice... Protocol 3 Based on Hash Function

A: Sign

```
s=D(f(m),kA[priv]) //sign digest
c=C(m,kB[pub])     //encrypt
send <A,c,s>
```

B: Verify

```
m*=D(c,kB[priv])       //decrypt
if f(m*)=C(s,kA[pub]) //verify
    then yes else no
```


Comments

- A signed message can be *replayed* at a later time: “Transfer $100 from *A* to *B*’s account”
- Need to add *timestamps*
- How do *A* and *B* obtain each other’s public keys?
- Simple minded message exchange subject to “man-in-the-middle” attack

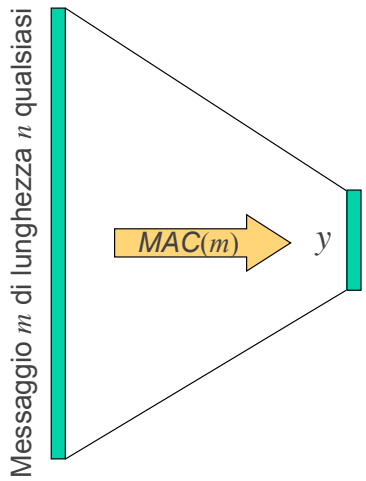
Autenticazione

- Identificazione: stabilire l'identità di un utente
- Autenticazione: qualcosa in più di identificazione
- Mittente spedisce un messaggio a destinatario
  Destinatario autentica il messaggio se:
  - identifica mittente
  - verifica l'integrità del messaggio (messaggio non modificato)

Message Authentication Code (MAC)

- Immagine breve (e di lunghezza fissa) del messaggio che può essere generata da un solo mittente conosciuto dal destinatario
- Ottenuto attraverso una chiave segreta condivisa tra il mittente e il destinatario
- Può essere usata per *autenticare* il messaggio: *identificazione* e *integrità*
- Simile a firme digitali ma senza la proprietà "non repudiation" (la destinazione può "firmare" qualunque messaggio)

MAC



y ha una lunghezza fissa, indipendente da n

Esempio di MAC

- Otteniamo un MAC applicando una funzione hash alla concatenazione di *m* e della chiave *k*
- MAC non è invertibile. Può solo essere calcolato di nuovo ma non invertito!!!

Esempio di Autenticazione usando MAC

- A spedisce $(m, f(mk))$ a B
- B riceve (μ, ω)
- B conosce k quindi può calcolare $f(\mu k)$
- B confronta $f(\mu k)$ con ω
- Se $f(\mu k) = \omega$, B conclude che $\mu = m$ (integro) e che il mittente di m è effettivamente A

Perché ?

- L'intruso X non può spedire un messaggio a *B* fingendosi *A* perché non conosce k

$\Rightarrow$ IDENTIFICAZIONE

- L'intruso X intercettando $(m, f(mk))$ non può modificare m perché non conosce k

$\Rightarrow$ INTEGRITA'

Autenticazione + Confidenzialità

- A spedisce $(C_k(m), f(mk))$ a B
- B riceve (μ, ω)
- B conosce k quindi decodifica μ

  $D_k(\mu)$ e risale a m^*
- B conosce k quindi può calcolare $f(m^*k)$
- B confronta $f(m^*k)$ con ω
- Se $f(m^*k) = \omega$, B conclude che $m^* = m$ (integro) e che il mittente di m è effettivamente A